Gossip and Attend: Context-Sensitive Graph Representation Learning

Zekarias T. Kefato, Sarunas Girdzijauskas
KTH Royal Institute of Technology
School of Electrical Engineering and Computer Science
Kista, Stockholm
{zekarias,sarunasg}@kth.se

Abstract

Graph representation learning (GRL) is a powerful technique for learning low-dimensional vector representation of high-dimensional and often sparse graphs. Most studies explore the structure and metadata associated with the graph using random walks and employ an unsupervised or semi-supervised learning schemes. Learning in these methods is context-free, resulting in only a single representation per node. Recently studies have argued on the adequacy of a single representation and proposed context-sensitive approaches, which are capable of extracting multiple node representations for different contexts. This proved to be highly effective in applications such as link prediction and ranking.

However, most of these methods rely on additional textual features that require complex and expensive RNNs or CNNs to capture high-level features or rely on a community detection algorithm to identify multiple contexts of a node.

In this study we show that in-order to extract high-quality context-sensitive node representations it is not needed to rely on supplementary node features, nor to employ computationally heavy and complex models. We propose GOAT, a context-sensitive algorithm inspired by gossip communication and a mutual attention mechanism simply over the structure of the graph. We show the efficacy of GOAT using 6 real-world datasets on link prediction and node clustering tasks and compare it against 12 popular and state-of-the-art (SOTA) baselines. GOAT consistently outperforms them and achieves up to 12% and 19% gain over the best performing methods on link prediction and clustering tasks, respectively.

Introduction

GRL is a powerful tool for learning the representation of a graph. Such a representation gracefully lends itself to a wide variety of network analysis tasks, such as link prediction, node clustering, node classification, recommendation, etc.

Naturally, users in real world networks belong to multiple contexts at a time. For instance, on interaction networks such as YouTube users usually interact (watch, like, comment and so on) with videos from different categories or topics depending on their interest. On social networks, like Facebook users tend to befriend others from multiple aspects as a result of communication over different contexts (e.g. country, school, religion, work and so on). This property is prevalent in many other areas, such as e-commerce, drug-target interaction networks and so on.

However, in most GRL studies, the learning is oblivious to such contexts (context-free) (Perozzi, Al-Rfou, and Skiena 2014; Grover and Leskovec 2016; Wang, Cui, and Zhu 2016; Perozzi, Kulkarni, and Skiena 2016; Yang et al. 2015; Pan et al. 2016; Sheikh, Kefato, and Montresor 2019). This is to say that all the context information is squeezed into a single (global) latent representation. In many cases, particularly for sparse graphs this leads to the loss of important details, and hence decreased performance in network analysis tasks.

Recently, a complementary line of research has questioned the adequacy of single representations per node and pursued a context-sensitive approach (Epasto and Perozzi 2019; Tu et al. 2017; Zhang et al. 2018; Kefato and Girdzijauskas 2020). This approach learns multiple representations per node to capture the different contexts that a node is part of. That is, given an anchor node, its representation changes depending on another target (context) node it is coupled with. A context node can be sampled from a neighborhood (Tu et al. 2017; Zhang et al. 2018), community affiliations (Epasto and Perozzi 2019), random walk (Ying et al. 2018), and so on. In this study we sample from a node neighborhood (nodes connected by an edge). That is, different neighbors of a given node potentially provide its multiple contexts. Thus, in the learning process of our approach representation of a source node changes depending on the target (context) node it is accompanied by. Studies have shown that context-sensitive approaches significantly outperform previous context-free SOTA methods in link-prediction task. A related notion (Peters et al. 2018; Devlin et al. 2018) in NLP has significantly improved SOTA across several NLP tasks.

In this study we propose GOAT[1] (**Go**ssip and **At**tend), a context-sensitive graph representation learning algorithm that is inspired by *gossip communication protocol* and *multi-way attention mechanism* (dos Santos et al. 2016). In order to facilitate understanding of nodes' context, GOAT allows

[1]Source code: https://github.com/zekarias-tilahun/goat

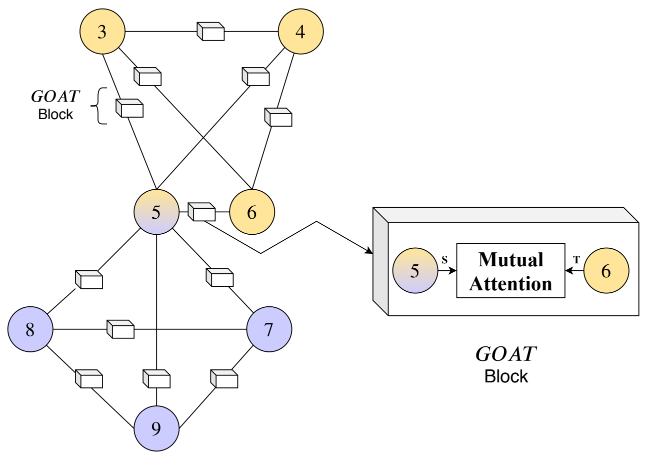


Figure 1: Left, a graph with a set of nodes having soft affiliation to groups indicated by colors and the cubes on the edges are parallel execution blocks of a shared GOAT model. Right, an illustration of a GOAT block for gossip partners 5 and 6

.

each node to gossip with each neighbor in its surrounding (context) in parallel, like the gossip communication protocol. To this end, a node uses its neighborhood as a message to be sent to the gossiping partners. Then, through a mutual (multi-way) attention mechanism, nodes will be allowed to learn the context that they are part of by cross examining their neighborhood against the message they received.

For example, in Fig. 1 when node 5 gossips with node 6, they both examine the message from the other node, i.e. the neighborhood set, which are $\{3, 4, 6, 8, 9, 7\}$ and $\{3, 4, 5\}$ respectively. In the message exchange, we want these nodes to understand that they have a shared neighborhood due to nodes 3 and 4 using the mutual-attention mechanism. That is, we seek that both nodes, 5 and 6, pay more attention to 3 and 4 and little attention to and/or ignore the other ones. On the other hand, when node 5 gossips with another node, e.g. node 7, we want the attention to shift to nodes 8 and 9. Therefore, each time a node gossips with another context node, it pays attention to different part of its neighborhood depending on its gossip partner. The intuition behind understanding neighborhood is reflected by changing the latent representation of a node depending on with whom it is gossiping with. This in turn enables us to learn multiple representations per node that capture multiple facets of the node instead of just one.

Note, while GOAT is inspired by gossip communication, it is not a decentralized algorithm as we monitor a central state (global embedding of nodes) and requires synchronisation.

GOAT

GOAT works over a graph $G = (V, E)$ with a set of n nodes V and m edges E. G can be directed or undirected and weighted or unweighted. Without loss of generality we assume that G is a weighted directed graph. For any node $v \in V$, $D(v) = |\{u : (u, v) \in E \vee (v, u) \in E\}|$ denotes its degree and for every directed edge (s, t), w_{st} denotes the weight associated to the edge.

The need for the gossip-like communication between pairs of nodes stems from our objective of learning context-sensitive (multiple) representations of nodes. That is, by allowing nodes to independently communicate (gossip) with their neighbors we enable them to identify/understand the multiple contexts that they are part of. For example in Fig. 1, after a set of parallel gossips between 5 and other nodes in two of its contexts we want node 5 to know that it is part of two contexts as indicated by the colors of the nodes.

However, in GOAT, similar to existing studies (Tu et al. 2017; Zhang et al. 2018), the number of representations per node is the same as the number of gossip partners that it has, for example 6 representations for node 5. This is because each gossip partner provides an understanding of a specific context that needs to be reflected by the representations. For this reason, one has to ensure that representations of a node within a specific context are very similar to each other. Thus, we use a mutual-attention mechanism to ensure that such representations are close to each other. After training, one can employ nearly constant-time algorithms like *locality sensitive hashing* (LSH) to collapse multiple representations of a node within the same context into a single one (Kumar, Zhang, and Leskovec 2019).

Gossiping in GOAT: In the gossip-like communication that we seek to establish between a pair of nodes $(u, v) \in E$, the exchanged message is a neighborhood function $f_n : V \times \mathbb{Z} \to 2^V$, which maps each node $u \in V$ to a set of N nodes $N_u \subseteq V$ sampled from the neighborhood of u. A simple way of materializing N_u is by sampling (without replacement) from the first-order neighbors of u, that is,

$$N_u = f_n(u, N) = [v : (u, v) \in E \vee (v, u) \in E] \quad (1)$$

where $N = |N_u|$ and for the i^{th} neighbor v, $N_u[i] = v, \nexists j \neq i$, where $N_u[j] = v$. A natural assumption that we have on N_u is that the order of nodes in N_u has no intrinsic meaning. In addition, though one can explore more sophisticated neighborhood sampling functions, in this study we simply consider the first order neighborhood.

Instead of the identity of its neighbors, $u_i \in N_u$, node u uses a global embedding $\boldsymbol{E}_{u_i}$ of u_i to communicate with its gossip partners. The matrix $\boldsymbol{E}$ defines the global (context-free) embedding of nodes and $\boldsymbol{E}_v$ denotes the global embedding of any node $v \in V$.

Therefore given a source s and a target t node, where $(s, t) \in E$, the actual messages that node s and t use to gossip are encoded using $\boldsymbol{S}$ and $\boldsymbol{T}$, respectively.

$$\boldsymbol{S} = [\boldsymbol{E}_u : u \in N_s], \quad \boldsymbol{T} = [\boldsymbol{E}_v : v \in N_t],$$

where $\boldsymbol{S} \in \mathbb{R}^{d \times N}$, $\boldsymbol{T} \in \mathbb{R}^{d \times N}$, and d is the embedding dimension.

Mutual Attention As we have alluded in an earlier discussion we use the attention mechanism so that a pair of nodes can mutually instill understanding of the shared context.

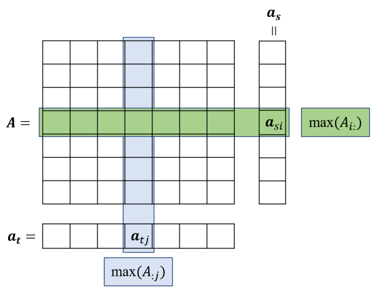


Figure 2: The alignment matrix $\boldsymbol{A}$ and the column-wise max pooling, highlighted by the green box $\boldsymbol{a}_{si} = \max(\boldsymbol{A}_{i:})$, and row-wise max-pooling operations, highlighted by the blue box $\boldsymbol{a}_{sj} = \max(\boldsymbol{A}_{:j})$ to compute the unnormalized attention weight vectors $\boldsymbol{a}_s$ and $\boldsymbol{a}_t$ of nodes s and t respectively.

The mutual attention works over the messages from the gossiping nodes $(s,t) \in E$. Recall the messages are specified by the two d–by–N matrices $\boldsymbol{S}$ and $\boldsymbol{T}$, where each column is a global embedding of the neighbors in N_s and N_t of s and t, respectively. In a nutshell, our strategy is to use an aggregated global embedding of the "important" neighbours to infer a context-sensitive embedding. Importance is quantified based on attention weights of neighbor nodes. A neighbor's weight will be learned depending on how much information it has contained regarding the shared context of the gossipers.

Formally, we achieve this by first computing a pair-wise soft alignment score between nodes in the set $N_s \times N_t$ as

$$\boldsymbol{A} = \boldsymbol{S}^T \cdot \boldsymbol{T} \tag{2}$$

Matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is a square matrix, where the vector in the i^{th} row $\boldsymbol{A}_{i:} \in \mathbb{R}^N$ is associated with the i^{th} neighbor $u_i \in N_s$ of s. Each of the components $\boldsymbol{A}_{ij}$ of $\boldsymbol{A}_{i:}$ encode how much node u_i's global embedding $\boldsymbol{S}^T_{u_i} = \boldsymbol{E}_{u_i}$ is similar/aligned to the global embeddings of each of the neighbors $[\boldsymbol{E}_{v_j} : v_j \in N_t]$ of t. Similarly, the vector in the j^{th} column $\boldsymbol{A}_{:j} \in \mathbb{R}^N$ is associated with the j^{th} neighbor $v_j \in N_t$ of t. The components of the vector are soft alignment scores between the global embedding $\boldsymbol{T}_{v_j} = \boldsymbol{E}_{v_j}$ of v_j and the global embedding of all the neighbors $[\boldsymbol{E}_{u_i} : u_i \in N_s]$ of s.

Therefore, if we inspect the maximum value $\boldsymbol{a}_{si} = \max(\boldsymbol{A}_{i:})$ of a particular row $\boldsymbol{A}_{i:}$ associated with a neighbor $u_i \in N_s$ (the green box in Fig. 2), that will be the maximum alignment score between u_i and neighbors $v_j \in N_t$ of t. Thus, by examining the maximum value from all the neighbors of s as $max([\boldsymbol{a}_{si} = \max(\boldsymbol{A}_{i:}) : u_i \in N_s])$ one can tell which neighbor of s is maximally aligned, *i.e.*, provide information on the shared context between s and t. So, to identify important neighbors N_s of s that align with the neighbors N_t of its gossip partner t, we perform a column-wise max-pooling operation on $\boldsymbol{A}$ and obtain an attention weight vector $\boldsymbol{a}_s \in \mathbb{R}^N$ as in Eq. 3. A similar inspection can be done on t as shown by the blue box in Fig. 2, and ultimately attention weight vector $\boldsymbol{a}_t$ of t can be computed by doing a row-wise max-pooling using Eq. 4.

$$\boldsymbol{a}_s = [\max(\boldsymbol{A}_{i:}) : u_i \in N_s] \tag{3}$$

$$\boldsymbol{a}_t = [\max(\boldsymbol{A}_{:j}) : v_j \in N_t] \tag{4}$$

We expect the attention weights of the important nodes, which are in the shared context of s and t, to have higher weights and the rest to have very small weights. Thus, if we take the weighted sum of the neighbors global-embedding, the global embedding from the important neighbors will have a stronger impact than the less important ones. This is exactly what we do to compute a context-sensitive representations of the gossipers. More formally, we compute the context-sensitive representations $\boldsymbol{r}_s \in \mathbb{R}^{d \times 1}$ and $\boldsymbol{r}_t \in \mathbb{R}^{d \times 1}$ of s and t, respectively, as the weighted sum of their neighbors global embedding using Equations 5 and 6.

$$\boldsymbol{r}_s = \boldsymbol{S} \cdot \texttt{softmax}(\boldsymbol{a}_s)^T \tag{5}$$

$$\boldsymbol{r}_t = \boldsymbol{T} \cdot \texttt{softmax}(\boldsymbol{a}_t)^T \tag{6}$$

Softmax is used to normalize the attention weights. Once we devise a learning objective, the attention weights in $\boldsymbol{a}_s$ and $\boldsymbol{a}_t$ should enable us to effectively distinguish important neighbors.

Optimization Objective: In order to train the aforementioned model, we employ the most commonly used learning objective in unsupervised GRL. That is, to maximize the likelihood of the graph or the observed edges, E. Concretely, we optimize the weighted negative log-likelihood of the edges specified in Eq. 7.

$$\mathcal{L}(E) = \min -\frac{1}{|E|} \sum_{(s,t) \in E} w_{st} \log P(t|s) \tag{7}$$

Equation 7 seeks to minimize the negative log-likelihood of observing node t as the gossip partner given node s, and $P(t|s)$ is estimated using the softmax formulation as follows

$$P(t|s) = \frac{\exp(\boldsymbol{r}_s \cdot \boldsymbol{r}_t)}{\sum_{w \in V} \exp(\boldsymbol{r}_s \cdot \boldsymbol{r}_w)} \tag{8}$$

However, due to the normalization constant that should be computed each time a node changes a gossip partner, Eq. 8 is expensive to compute and we resort to negative sampling (Mikolov et al. 2013). Negative nodes are sampled from the distribution P_V, and a couple of alternatives, such as, the uniform and unigram distributions have been tested and (Mikolov et al. 2013) reported that the unigram distribution raised to the power of 0.75 significantly outperforms the others, and hence we sample negative nodes according to the empirical distribution $P_V(u) = \frac{D(u)^{0.75}}{n}$.

Dataset	#Nodes	#Edges	Features
Cora	2277	5214	Paper Abstract
Cora2	2708	5278	Paper Abstract
Citeseer	3327	4676	Paper Abstract
Pubmed	19717	44327	Paper Abstract
Zhihu	10000	43894	User post
Email	1005	25571	NA

Table 1: Summary of datasets, the *Features* column is relevant to some of the baselines not GOAT.

Dataset	ρ	N	λ	d
Cora (2)	0.5	100	0.0001	200
Citeseer				
Pubmed				
Email	0.8			
Zhihu	0.65	250		

Table 2: Configuration of GOAT, ρ (dropout rate), N (neighborhood size), λ (learning rate), and d (dimension of the latent representation).

Computational Complexity: The learning in GOAT is mainly affected by the number of edges, $\mathcal{O}(m)$. The first task is to compute the embedding of each neighbor of the gossiping nodes s and t. Since just a lookup is required to compute the embedding $\boldsymbol{E}_u$ of any neighbor u, the cost required to embed all neighbors of the gossipers using $\boldsymbol{S}$ and $\boldsymbol{T}$ is $\mathcal{O}(2N)$, assuming that lookup is a constant time operation. Second, the attention step involves a matrix multiplication given in Eq. 2, which has a cost of $\mathcal{O}(2N^2d)$. Therefore, the asymptotic computational cost of GOAT is proportional to $\mathcal{O}(m(N^2d + 2N))$. However, N and d are usually very small (less than 300) and in addition one can capitalize on highly specialized linear algebra and machine learning libraries [2]. Hence, GOAT can easily scale to large graphs, as it is mainly affected by the number of edges. Furthermore, its design makes it easy to parallelize or decentralize the implementation.

Empirical Results

In this section we provide an empirical evaluation of GOAT. To this end, we carried out experiments using the following six datasets, and a basic summary is given in Table 1.

1. Three of the datasets (Cora, Citeseer, and Pubmed) (Tu et al. 2017; Zhang et al. 2018; Kipf and Welling 2016): are citation network datasets, where a node represents a paper and an edge $(u, v) \in E$ represents that paper u has cited paper v. For Cora we use two versions, and we refer to them as Cora and Cora2.
2. Zhihu (Tu et al. 2017; Zhang et al. 2018): is the biggest social network for Q&A and it is based in China. Nodes are the users and the edges are follower relations between the users.
3. Email (Leskovec, Kleinberg, and Faloutsos 2007): is an email communication network between the largest European research institutes. A node represents a person and an edge $(u, v) \in G$ denotes that person u has sent an email to v.

[2]We use the Numpy and PyTorch toolkits to implement GOAT

Datasets under 1 and 2 have features (documents) associated to nodes. Some of the baselines, discussed beneath in the 2 and 3 category, require textual information, and hence they consume the aforementioned features. The Email dataset has ground-truth community assignment for nodes based on a person's affiliation to one of the 42 departments.

We compare our method against the following 12 popular and SOTA baselines grouped as:

1. *Structure based methods*: DEEPWALK (Perozzi, Al-Rfou, and Skiena 2014), NODE2VEC (Grover and Leskovec 2016), WALKLETS (Perozzi, Kulkarni, and Skiena 2016), ATTENTIVEWALK (Abu-El-Haija et al. 2017), LINE (Tang et al. 2015).
2. *Structure & content based methods*: TRIDNR (Pan et al. 2016), TADW (Yang et al. 2015), CENE (Sun et al. 2016).
3. *Structure & content based Context-sensitive methods*: CANE (Tu et al. 2017), DMTE (Zhang et al. 2018).
4. *Structure based context-sensitive method*: SPLITTER (Epasto and Perozzi 2019).
5. *GCN based method*:VGAE (Kipf and Welling 2016).

Note that the closest algorithm to GOAT is SPLITTER, not because of the algorithmic design but because both of them are topology (structure) based context-sensitive methods. We also include a variant of GOAT called GOATGLOBAL that uses the global embedding of the nodes. Experiments are carried out on two tasks, which are link prediction and node clustering, all of them are performed using a 24-Core CPU and 125GB RAM Ubuntu 18.04 machine.

Link Prediction

Link prediction is an important task that graph embedding algorithms are applied to. Particularly context-sensitive embedding techniques have proved to be well suited for this task. Similar to existing studies we perform this experiment using a fraction of the edges as a training set. We hold out the remaining fraction of the edges from the training phase and we will only reveal them during the test phase, results are reported using this set. All hyper-parameter tuning is performed by taking a small fraction (20%) of the training set as a validation set.

Setup: In-line with existing techniques (Tu et al. 2017; Zhang et al. 2018), the percentage of training edges ranges from 15% to 95% by a step of 10. The hyper-parameters of all algorithms are tuned using random-search. For some of the baselines, our results are consistent with what is reported in previous studies, and hence for Cora and Zhihu we simply report these results.

Except the "unavoidable" hyper-parameters (eg. learning rate, regularization/dropout rate) that are common in all the algorithms, our model has just one hyper-parameter, which

Dataset	Algorithm	% of training edges								
		15%	25%	35%	45%	55%	65%	75%	85%	95%
Cora	DeepWalk	56.0	63.0	70.2	75.5	80.1	85.2	85.3	87.8	90.3
	Line	55.0	58.6	66.4	73.0	77.6	82.8	85.6	88.4	89.3
	Node2Vec	55.9	62.4	66.1	75.0	78.7	81.6	85.9	87.3	88.2
	WalkLets	69.8	77.3	82.8	85.0	86.6	90.4	90.9	92.0	93.3
	AttentiveWalk	64.2	76.7	81.0	83.0	87.1	88.2	91.4	92.4	93.0
	tadw	86.6	88.2	90.2	90.8	90.0	93.0	91.0	93.0	92.7
	TriDnr	85.9	88.6	90.5	91.2	91.3	92.4	93.0	93.6	93.7
	cene	72.1	86.5	84.6	88.1	89.4	89.2	93.9	95.0	95.9
	cane	86.8	91.5	92.2	93.9	94.6	94.9	95.6	96.6	97.7
	dmte	91.3	93.1	93.7	95.0	96.0	97.1	97.4	**98.2**	**98.8**
	splitter	65.4	69.4	73.7	77.3	80.1	81.5	83.9	85.7	87.2
	GoatGlobal	93.3	95.4	96.2	97.1	97.4	**97.6**	97.5	98.0	98.3
	Goat	**96.7**	**96.9**	**97.1**	**97.5**	**97.6**	**97.6**	**97.8**	98.0	98.2
	GAIN%	5.4%	3.8%	3.4%	2.5%	1.6%	0.5%	0.4%		
Zhihu	DeepWalk	56.6	58.1	60.1	60.0	61.8	61.9	63.3	63.7	67.8
	Line	52.3	55.9	59.9	60.9	64.3	66.0	67.7	69.3	71.1
	Node2Vec	54.2	57.1	57.3	58.3	58.7	62.5	66.2	67.6	68.5
	WalkLets	50.7	51.7	52.6	54.2	55.5	57.0	57.9	58.2	58.1
	AttentiveWalk	69.4	68.0	74.0	75.9	76.4	74.5	74.7	71.7	66.8
	tadw	52.3	54.2	55.6	57.3	60.8	62.4	65.2	63.8	69.0
	TriDnr	53.8	55.7	57.9	59.5	63.0	64.2	66.0	67.5	70.3
	cene	56.2	57.4	60.3	63.0	66.3	66.0	70.2	69.8	73.8
	cane	56.8	59.3	62.9	64.5	68.9	70.4	71.4	73.6	75.4
	dmte	58.4	63.2	67.5	71.6	74.0	76.7	78.7	80.3	82.2
	splitter	59.8	61.5	61.8	62.1	62.1	62.4	61.0	60.7	58.6
	GoatGlobal	66.1	74.6	74.1	75.2	73.2	68.8	71.1	73.6	74.7
	Goat	**82.2**	**80.7**	**82.3**	**82.4**	**85.1**	**85.3**	**84.5**	**84.4**	**83.7**
	GAIN%	12.8%	12.7%	8.3%	6.5%	8.7%	8.9%	5.8%	4.1%	1.5%
Email	DeepWalk	69.2	71.4	74.1	74.7	76.6	76.1	78.7	75.7	79.0
	Line	65.6	71.5	73.8	76.0	76.7	77.8	78.5	77.9	78.8
	Node2Vec	66.4	68.6	71.2	71.7	72.7	74.0	74.5	74.4	76.1
	WalkLets	70.3	73.2	75.2	78.7	78.2	78.1	78.9	80.0	78.5
	AttentiveWalk	68.8	72.5	73.5	75.2	74.1	74.9	73.0	70.3	68.6
	splitter	69.2	70.4	69.1	69.2	70.6	72.8	73.3	74.8	75.2
	GoatGlobal	78.6	80.3	80.8	81.1	81.3	81.8	82.0	82.1	82.6
	Goat	**78.9**	**81.0**	**81.2**	**81.4**	**81.7**	**82.4**	**82.3**	**82.6**	**83.1**
	GAIN%	8.3%	5.6%	3.4%	0.8%	1.8%	2.7%	2.4%	1.5%	1.7%

Table 3: AUC results for the link prediction task on the Cora, Zhihu, and Email datasets.

is the neighborhood size – N, and for nodes with smaller neighborhood size we use zero padding. As we shall verify later, Goat is not significantly affected by the choice of this parameter.

The quality of the link prediction is measured using the area under the receiver operating characteristic curve (AUC) and average precision (AP) scores. AUC indicates the probability that a randomly selected pair $(u, w) \notin E$ will be ranked lower than an edge $(u, v) \in E$ in the test set. The AP indicates the quality of the overall ranking as a summary of precision and recall scores at different thresholds. Rank of a pair of nodes is computed as the dot product of their representation. For all the algorithms the representation size – d is 200 and Goat's configuration is shown in Table 2.

Results: The results for Cora, Zhihu, and Email datasets are reported in Table 3. Goat outperforms the SOTA baselines in all cases for Zhihu and Email, and in almost all cases for Cora. One can see that as we increase the percentage of training edges, performance significantly increases for all the baselines. As indicated by the "Gain" row, Goat achieves up to 12.8% improvement over SOTA and context-sensitive techniques. Notably the gain is pronounced for smaller values of percentage of edges used for training. This shows that Goat is suitable both in cases where there are several missing links and most of the links are present.

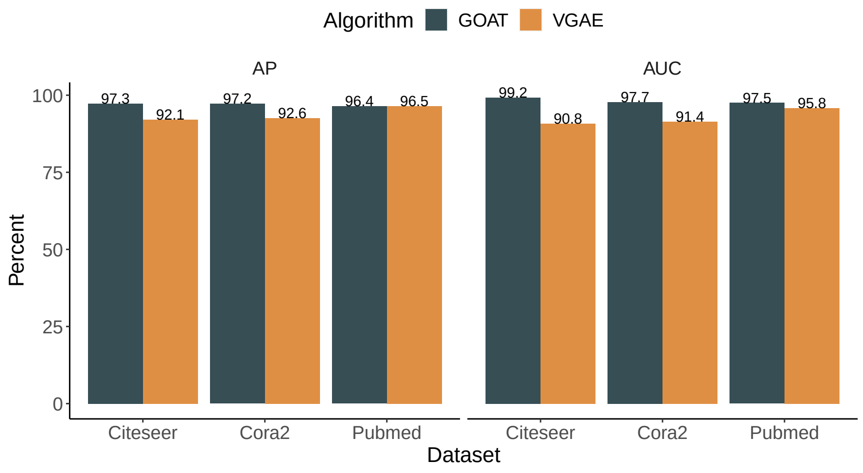


Figure 3: AUC and AP results of VGAE and GOAT for link prediction task.

Recently, several studies have pursued a type of GRL models known as graph convolutional neural networks (GCNs) (Kipf and Welling 2017; Velickovic et al. 2017; Abu-El-Haija et al. 2019; Hamilton, Ying, and Leskovec 2017; Wu et al. 2019; Kipf and Welling 2016). Though most of them are widely used for semi-supervised node classification, in this study we compare GOAT with a type of GCN, called variational graph auto-encoder (VGAE) that is commonly used for the link-prediction task (Kipf and Welling 2016; Schlichtkrull et al. 2017) using three of the remaining datasets from the VGAE paper. For a fair comparison, we use exactly the same configuration, that is, the same training (90%) and test (10%) sets provided by the authors.

In Fig. 3 we report the AUC and AP empirical results on Citeseer, Cora2 and Pubmed datasets, yet again we show that GOAT outperforms VGAE in almost all the cases, by upto 8%. A consistent observation that we have in the above results is that, compared to all the baselines GOAT's performance is robust even when we have little observation.

Node Clustering

Nodes in a network has the tendency to form cohesive structures based on shared latent properties. These structures are commonly known as groups, clusters or communities and their identification has important real-world applications. We use the Email dataset since it has 42 ground truth communities. Recall that this dataset has only structural information, thus we have included structure-based methods only.

Setup: Since each node belongs to exactly one cluster, we employ the k-Means algorithm to identify clusters. The learned representations of nodes by a certain algorithm are the input features of the clustering algorithm. In this experiment the percentage of training edges varies from 35% to 95% by a step of 20%, for the rest we use the same configuration as in the above experiment.

Given the ground truth community assignment y of nodes and the predicted community assignments $\hat{y}$, usually the agreement between y and $\hat{y}$ are measured using mutual information $I(y, \hat{y})$. However, I is not bounded and difficult for comparing methods, hence we use two other variants of I (Vinh, Epps, and Bailey 2010). Which are, the normalized mutual information $NMI(y, \hat{y})$, which simply normalizes I and adjusted mutual information $AMI(y, \hat{y})$, which adjusts or normalizes I to random chances.

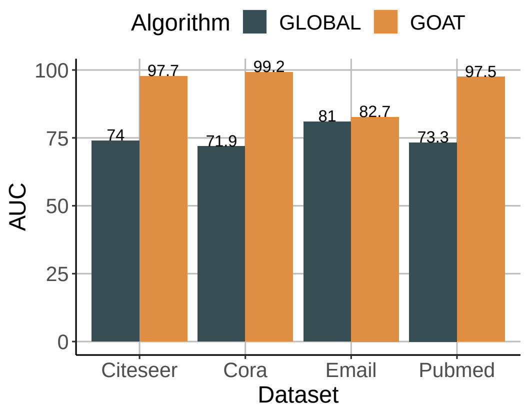


Figure 4: Comparison between GOAT and GLOBAL.

Results: The results of this experiment are reported in Table 4, and GOAT significantly outperforms all the baselines by up to 19% with respect to AMI score. Consistent to our previous experiment GOAT's performance is not affected by the change in the percentage of the training edges for both NMI and AMI.

Ablation Study

To appreciate the importance of the mutual-attention component of GOAT, we carry out an experiment by removing the attention component. That is, instead of the context-sensitive

Algorithm	**%of training edges**							
	35%		**55%**		**75%**		**95%**	
	NMI	**AMI**	**NMI**	**AMI**	**NMI**	**AMI**	**NMI**	**AMI**
DEEPWALK	41.3	28.6	53.6	44.8	50.6	42.4	57.6	49.9
LINE	44.0	30.3	49.9	38.2	53.3	42.6	56.3	46.5
NODE2VEC	46.6	35.3	45.9	35.3	47.8	38.5	53.8	45.5
WALKLETS	47.5	39.9	55.3	47.4	54.0	45.4	50.1	41.6
ATTENTIVEWALK	42.9	30.0	45.7	36.5	44.3	35.7	47.4	38.5
SPLITTER	38.9	23.8	43.2	30.3	45.2	33.6	48.4	37.6
GOAT	**66.5**	**57.2**	**65.6**	**56.7**	**66.4**	**57.9**	**65.5**	**57.0**
%Gain	19%		10.3%		12.4%		7.9%	

Table 4: NMI and AMI scores for node clustering experiment on the Email dataset. The Gain is with respect to the NMI only.

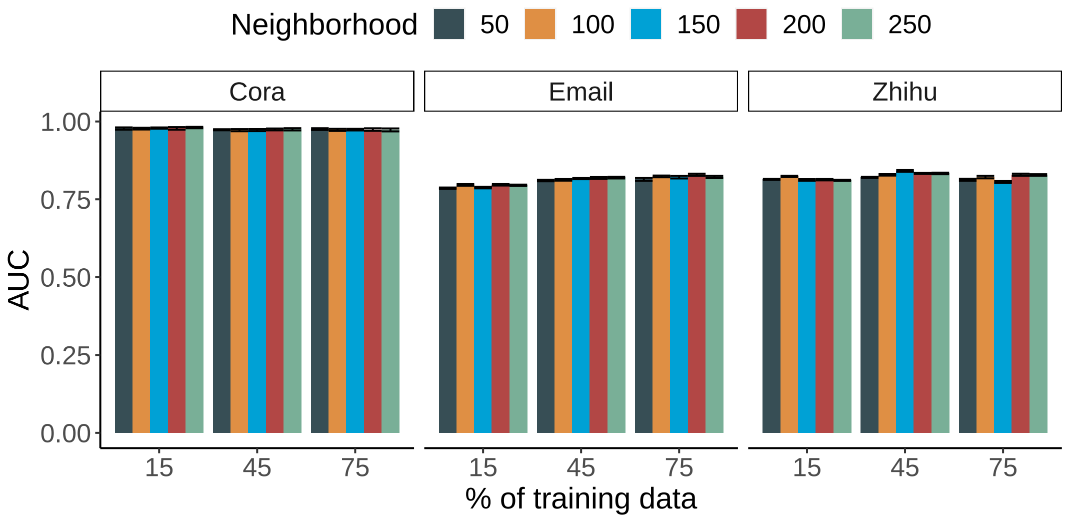


Figure 5: Sensitivity of GOAT to the size of node's neighborhood (N) on the link prediction task.

representations $\boldsymbol{r}_s$ and $\boldsymbol{r}_t$, in Eq. 8 we use the global embeddings $\boldsymbol{E}_s$ and $\boldsymbol{E}_t$, and refer to this variant simply as GLOBAL. It is similar to the second-order preserving variant of LINE (Tang et al. 2015). From Fig. 4, one can clearly see that the mutual-attention component of GOAT is crucial for its effectiveness.

Parameter Sensitivity Analysis

Now, we turn into analyzing the effect of the main hyperparameter of GOAT, which is the size of the neighborhood (N). In Fig 5 we show the effect of this parameter across different rate of training edges on the link prediction task. We observe that, regardless of the percentage of training edges, GOAT is not significantly affected by the change in N.

Scalability and Convergence

To empirically substantiate GOAT's scalability, we carry out experiments on synthetic graphs up to millions of edges, which are generated using BarabásiAlbert model. Fig. 6(A) shows the run time (y–axis) needed to complete an epoch for graphs with different number of edges, 50K-2M (x –axis), and we note that GOAT can finish an epoch in $\approx$ 7 min for the graph with 2M edges. Moreover, once the model hyperparameters are fixed, we have empirically observed that for large and dense graphs GOAT requires small number of epochs to converge. Fig 6(B) shows this observation, and the y–axis indicates the number of epochs required for GOAT to converge on 15% training edges in-order to achieve the performance reported in Section .

Case Study: Les Misrables

To shed more light on the GOAT model, here we briefly analyze the learned attention weights of a graph based on the encounter relations between the characters in the Les Misrables novel (Knuth 1993). A visualization is shown in Fig 7, along with community affiliations (contexts) of nodes based on modularity classes, indicated by the colors. First, we pick two arbitrary nodes, which are nodes 25 and 55. In Fig 8, we observe that both 25 and 55 strongly pay attention to the shared neighbors. Then, we let 55 to gossip with another node, *i.e.* 48, in a different context and show the resulting attention weights in Fig 9. Now, for neighbors of 55 we observe that the attention weight is concentrated on those neighbors that had less attention in the previous gossip.

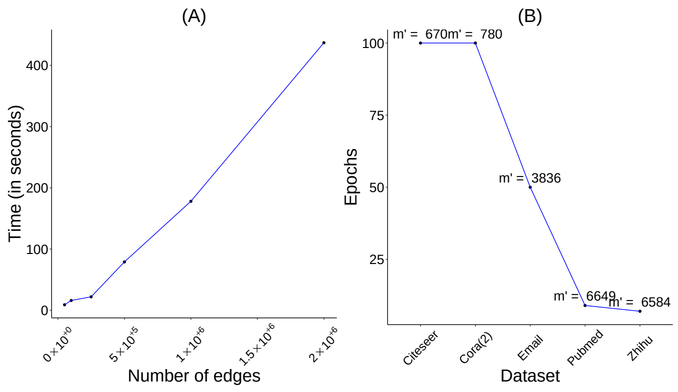


Figure 6: Run time (A) of GOAT on an NVIDIA QUADRO RTX 5000 GPU and its convergence (B). The annotation in (B) shows the fraction of training edges $m' = |E| \times .15$.

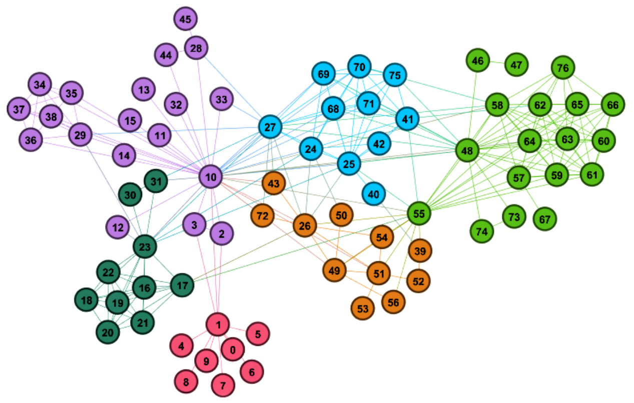


Figure 7: The Les Misrables character encounter graph with character's modulality class indicated by colors.

Related Work

Graph Representation Learning is usually carried out by exploring the structure of the graph and meta data, such as node attributes, attached to the graph (Perozzi, Al-Rfou, and Skiena 2014; Grover and Leskovec 2016; Tang et al. 2015; Perozzi, Kulkarni, and Skiena 2016; Wang, Cui, and Zhu 2016; Yang et al. 2015; Pan et al. 2016; Sheikh, Kefato, and Montresor 2019). Random walks are widely used to explore local/global neighborhood structures, which are then fed into a learning algorithm. Often, an unsupervised learning objective is specified using the maximum likelihood of neighboring nodes/attributes given a center node.

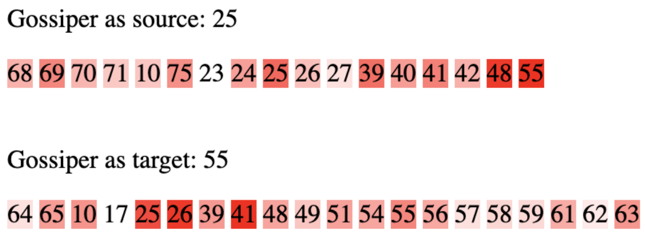
Recently, graph convolutional networks have also been proposed for semi-supervised network analysis tasks (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017; Wu et al. 2019; Velickovic et al. 2017; Abu-El-Haija et al. 2019). These algorithms are trained to learn different kinds of neighborhood feature aggregator functions using a downstream objective based on partial labels of nodes. All these methods are essentially different from our approach because they are context-free.

Context-sensitive learning is another paradigm for NRL that challenges the adequacy of a single representation of a node for applications such as, link prediction, product recommendation, ranking. While some of these methods (Tu et al. 2017; Zhang et al. 2018) rely on textual information, others have also shown that a similar goal can be achieved using just the structure of the graph (Epasto and Perozzi 2019). However, they require an extra step of persona decomposition that is based on microscopic level community detection algorithms to identify multiple contexts of a node. Besides, it is susceptible to errors propagating from wrong community assignments. Unlike the former approaches our algorithm does not require extra textual information and with respect to the later our approach does not require any sort of community detection algorithm.

Conclusion

In this study, we present a novel context-sensitive graph embedding algorithm called GOAT. GOAT is inspired by a gossip-like communication and mutual attention mechanism. Each node is allowed to gossip with each neighbor by using the remainder of the neighborhood as a message. By capitalizing on the mutual attention mechanism GOAT allows nodes to understand their contexts and infer multiple representations per node.

GOAT learns high-quality context-sensitive representations of nodes. We have empirically evaluated the quality of the representations and have shown that it consistently outperforms best performing SOTA context-sensitive and context-free baselines using 6 public datasets in link prediction and node clustering tasks, exhibiting significant improvements of up to 12% and 19% respectively.



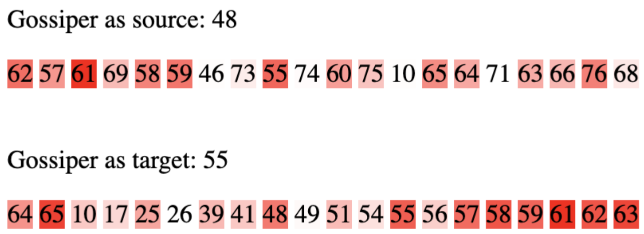
Gossiper as source: 25

68 69 70 71 10 75 23 24 25 26 27 39 40 41 42 48 55

Gossiper as target: 55

64 65 10 17 25 26 39 41 48 49 51 54 55 56 57 58 59 61 62 63

Figure 8: Visualization of the learned attention weights of neighbors of 25 and 55.



Gossiper as source: 48

62 57 61 69 58 59 46 73 55 74 60 75 10 65 64 71 63 66 76 68

Gossiper as target: 55

64 65 10 17 25 26 39 41 48 49 51 54 55 56 57 58 59 61 62 63

Figure 9: Visualization of the learned attention weights of neighbors of 48 and 55.

In a future work we seek to extend GOAT to a completely decentralized environment and investigate how node attributes can be integrated in the GOAT framework.

References

[Abu-El-Haija et al. 2017] Abu-El-Haija, S.; Perozzi, B.; Al-Rfou, R.; and Alemi, A. 2017. Watch your step: Learning graph embeddings through attention. *CoRR* abs/1710.09599.

[Abu-El-Haija et al. 2019] Abu-El-Haija, S.; Perozzi, B.; Kapoor, A.; Harutyunyan, H.; Alipourfard, N.; Lerman, K.; Steeg, G. V.; and Galstyan, A. 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. *CoRR*.

[Devlin et al. 2018] Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR* abs/1810.04805.

[dos Santos et al. 2016] dos Santos, C. N.; Tan, M.; Xiang, B.; and Zhou, B. 2016. Attentive pooling networks. *CoRR* abs/1602.03609.

[Epasto and Perozzi 2019] Epasto, A., and Perozzi, B. 2019. Is a single embedding enough? learning node representations that capture multiple social contexts. *CoRR* abs/1905.02138.

[Grover and Leskovec 2016] Grover, A., and Leskovec, J. 2016. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, 855–864. New York, NY, USA: ACM.

[Hamilton, Ying, and Leskovec 2017] Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *CoRR* abs/1706.02216.

[Kefato and Girdzijauskas 2020] Kefato, Z. T., and Girdzijauskas, S. 2020. Graph neighborhood attentive pooling.

[Kipf and Welling 2016] Kipf, T. N., and Welling, M. 2016. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*.

[Kipf and Welling 2017] Kipf, T. N., and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

[Knuth 1993] Knuth, D. E. 1993. *The Stanford GraphBase: A Platform for Combinatorial Computing*. New York, NY, USA: ACM.

[Kumar, Zhang, and Leskovec 2019] Kumar, S.; Zhang, X.; and Leskovec, J. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD 19*.

[Leskovec, Kleinberg, and Faloutsos 2007] Leskovec, J.; Kleinberg, J.; and Faloutsos, C. 2007. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data* 1(1).

[Mikolov et al. 2013] Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, 3111–3119. USA: Curran Associates Inc.

[Pan et al. 2016] Pan, S.; Wu, J.; Zhu, X.; Zhang, C.; and Wang, Y. 2016. Tri-party deep network representation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, 1895–1901. AAAI Press.

[Perozzi, Al-Rfou, and Skiena 2014] Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. *CoRR* abs/1403.6652.

[Perozzi, Kulkarni, and Skiena 2016] Perozzi, B.; Kulkarni, V.; and Skiena, S. 2016. Walklets: Multiscale graph embeddings for interpretable network classification. *CoRR* abs/1605.02115.

[Peters et al. 2018] Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. *CoRR* abs/1802.05365.

[Schlichtkrull et al. 2017] Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; van den Berg, R.; Titov, I.; and Welling, M. 2017. Modeling relational data with graph convolutional networks.

[Sheikh, Kefato, and Montresor 2019] Sheikh, N.; Kefato, Z. T.; and Montresor, A. 2019. gat2vec: Representation learning for attributed graphs. In *Journal of Computing*. Springer.

[Sun et al. 2016] Sun, X.; Guo, J.; Ding, X.; and Liu, T. 2016. A general framework for content-enhanced network representation learning. *CoRR* abs/1610.02906.

[Tang et al. 2015] Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. LINE: large-scale information network embedding. *CoRR* abs/1503.03578.

[Tu et al. 2017] Tu, C.; Liu, H.; Liu, Z.; and Sun, M. 2017. CANE: Context-aware network embedding for relation modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1722–1731. Vancouver, Canada: Association for Computational Linguistics.

[Velickovic et al. 2017] Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2017. Graph attention networks. *ArXiv* abs/1710.10903.

[Vinh, Epps, and Bailey 2010] Vinh, N. X.; Epps, J.; and Bailey, J. 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *J. Mach. Learn. Res.* 11:2837–2854.

[Wang, Cui, and Zhu 2016] Wang, D.; Cui, P.; and Zhu, W. 2016. Structural deep network embedding. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, 1225–1234. New York, NY, USA: ACM.

[Wu et al. 2019] Wu, F.; Zhang, T.; Jr., A. H. S.; Fifty, C.; Yu, T.; and Weinberger, K. Q. 2019. Simplifying graph convolutional networks. *CoRR* abs/1902.07153.

[Yang et al. 2015] Yang, C.; Liu, Z.; Zhao, D.; Sun, M.; and Chang, E. Y. 2015. Network representation learning with rich text information. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, 2111–2117. AAAI Press.

[Ying et al. 2018] Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W. L.; and Leskovec, J. 2018. Graph convolutional neural networks for web-scale recommender systems. *CoRR* abs/1806.01973.

[Zhang et al. 2018] Zhang, X.; Li, Y.; Shen, D.; and Carin, L. 2018. Diffusion maps for textual network embedding. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18. USA: Curran Associates Inc.