Performance of Common Classifiers on node2vec Network Representations

Mislav Pozek, Lucija Sikic, Petar Afric, Adrian S. Kurdija, Klemo Vladimir, Goran Delac, Marin Silic
University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, Croatia
mislav.pozek@fer.hr, lucija.sikic@fer.hr, petar.afric@fer.hr, adrian.kurdija@fer.hr, klemo.vladimir@fer.hr, goran.delac@fer.hr, marin.silic@fer.hr

Abstract - In this paper we evaluate the performance of different multi-class classifiers on network graphs. Since the node embedding techniques have been widely used to represent and analyze networks structures, we decide to transform network data (nodes and links) into attributes which are descriptive and contain correct information of its structure. For this purpose, we use a *state-of-the-art* algorithmic framework *node2vec*, which has been shown to outperform other popular methods when applied to multi-label classification as it manages to efficiently learn a mapping of nodes to a low-dimensional space of features. Applying this framework, we generate a set of representations for nodes of multiple network data sets. Using the generated representations, we evaluate the performance of common classifiers. We perform cross-validation and parameter tuning to get the best possible model of each classifier type. To compare their performance, we computed Precision, Recall and F1-score for each model on each data set. Following that, the obtained results are analyzed and compared.

I. INTRODUCTION

Networks are a universal language for describing and modeling complex systems in science, nature and technology. Since the data describing these systems has become available, network science researchers have developed various techniques to extract features from complex networks, which then can be used in various applications, such as detecting communities and predicting new friendships in social media or predicting the protein's functional labels. Although they have relied on user-defined heuristics to extract features, in recent years approaches have been devised that automatically learn to encode network structures into embeddings [2]. These network representation learning (NRL) approaches are based on deep learning and nonlinear dimensionality reduction. Introducing such generated features has led to state-of-the-art results in network-based tasks.

A challenging machine learning task in networks is the node classification problem. This problem has been proven to be useful in many real-world applications, especially in social network studies. The fact it requires not only network node attributes, but also the information of the linkage structure, shows it is more complex comparing to traditional classification problems. Additionally, in contrast to traditional classification, the network topology should be considered, as it contains the neighbors label information, which may be important and decisive.

Many techniques have been recently developed to generate representations of network nodes. These can be divided based on presence or absence of the node feature information in generating node embeddings. Techniques that use only network structure for that purpose try to find embeddings of nodes to low-dimensions so that similar nodes in the network have embeddings that are close together. The other group of techniques use both node features and network structure. It generates node embeddings by aggregating neighborhood information, which consist of node attributes and defined distance between nodes.

Since social networks attracts more and more research attention, we develop and analyze a procedure of multi-label classification on node embedding generated on this kind of network, using only the information extracted from the network structure. In our experiment we apply *node2vec* framework to generate node embeddings for Cora, Citeseer and Wiki social networks. The generated node embeddings are used to learn and evaluate common multi-class classifiers: ridge and lasso regression, nearest centroid and support-vector classifier [9]. Their performances are compared using well established measures: Precision, Recall and F1-score. The evaluation results obtained from conducted experiment strongly suggest that embeddings generated by *node2vec* framework can be successfully used as feature vectors for node classification algorithms.

The rest of the paper is organized as follows. Section II presents the related work. The algorithm *node2vec* is briefly described in section III, while the experiment is described in section IV. Obtained results from the experiment are presented in section V. Section VI concludes the paper.

II. RELATED WORK

Typical approaches to unsupervised network feature learning include some form of a network's matrix representation analysis. These methods are akin to dimensionality reduction techniques and are usually based on similar algorithms. Although several methods based on the network's matrix representation analysis were proposed (*e.g.*, Modularity Maximization [3] and TADW

[4]), such approaches do not scale well to larger networks due to the necessary step of matrix factorization.

Both edge modeling and deep learning methods have been proposed as an alternative to matrix factorization based methods. Edge modeling methods aspire to capture the network structure by modelling various probability distributions of vertex to vertex relations. An example is LINE [5], which models both joint and conditional probability distribution of connected vertices to capture the first and second-order proximity representations. These methods are severely restricted in encoding global network structures as they consider only observable vertex connectivity information.

Several unsupervised network representation learning models based on random walks have been proposed in recent years to alleviate this problem. DeepWalk [6] pioneered the approach by viewing random walks through graphs as sentences in a text corpus. DeepWalk applies a text representation learning method to learn generic representations of such walks and assign them to starting vertices. Although DeepWalk showed promising results, *node2vec;* a more flexible solution of exploring node neighborhoods using a biased random walk would prove to be better when applied across a wide range of different networks.

III. NODE2VEC ALGORITHM

In this section we define the *node2vec* algorithm along with the corresponding definitions from [1].

Node2vec is developed at Stanford University by Aditya Grover and Jure Leskovec in 2016. It is an efficient scalable algorithm for feature learning in networks. The algorithm is in accordance with established principles in network science, providing flexibility in discovering representations conforming to different equivalences.

For feature learning in networks in a semi-supervised manner the *node2vec* algorithm optimizes a custom graph-based objective function using SGD. The learned feature representations maximize the likelihood of preserving network neighborhood of nodes in a *d*-dimensional feature space. To generate network neighborhoods for nodes the 2nd order random walk approach is used.

The algorithm can be divided in three phases which are executed sequentially: preprocessing to compute transition probabilities, random walk simulations and optimization using SGD. Each phase is parallelizable and executed asynchronously, contributing to the overall scalability of the algorithm. In the subsections A, B and C the phases are briefly explained.

A. Computing transition probabilities

The first step of the *node2vec* algorithm is choosing an appropriate notion of a neighborhood, which allows learning representations that organize nodes based on their network roles and/or communities they belong to. This is achieved by developing a flexible biased random walk procedure that can explore neighborhoods in a BFS as well as DFS fashion.

They defined a 2nd order random walk with two parameters p and q which guide the walk. Parameter p controls the likelihood of immediately revisiting a node in the walk while parameter q allows the search to differentiate between "inward" and "outward" nodes. Both parameters allow search procedure to interpolate between BFS and DFS. Considering a random walk that just traversed edge (t, v) and now resides at node v, for decision on the next step the transition probabilities π_{vx} on edges (v, x) leading from v are evaluated. The unnormalized transition probability is set to $\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$, where

$$a_{pq} = \begin{cases} \frac{1}{p} & if \ \ d_{tx} = 0 \\ 1 & if \ \ d_{tx} = 1 \\ \frac{1}{q} & if \ \ d_{tx} = 2 \end{cases} \quad (1)$$

And d_{tx} denotes the shortest path distance between nodes t and x.

B. Simulation of random walks

Random walks provide a convenient mechanism to increase the effective sampling rate by reusing samples across different source nodes. By simulating the walk of length l, where $l > k$, k samples for $l - k$ nodes are generated at once due to the Markovian nature of random walk. Since the representations are learned for all nodes, the random walks of length l are simulated starting from every node. Formally, having the ith node in the walk denoted as c_i and starting with $c_0 = u$, nodes c_i are generated by the following distribution:

$$P(c_i = x | c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & if \ \ (v, x) \in E \\ 0 & otherwise \end{cases} \quad (2)$$

Where π_{vx} is the unnormalized transition probability between nodes u and x, and Z is the normalizing constant.

C. Optimization phase

The *node2vec* algorithm formulates the feature learning in networks as a maximum likelihood optimization problem. To solve this problem the authors extended the Skip-Gram architecture [10] for feature learning by randomized procedure that samples many different neighborhoods of a given source node u.

Formally, for every source node $u \in V$ of a given network $G = (V, E)$ a network neighborhood of node u generated through a neighborhood sampling strategy S, denoted as $N_S(u) \subset V$, is defined. Furthermore, let $f: V \rightarrow \mathbb{R}^d$ be the mapping function from nodes to features representations to be learnt, where *d* represents the number of dimensions of feature representations. To learn features the following objective function, which maximizes the log-probability observing a network neighborhood $N_S(u)$ for a node u conditioned on its feature representation *f*, is optimized:

$$\max_f \sum_{u \in V} \log Pr(N_S(u) | f(u)) \quad (3)$$

By taking the assumptions of the conditional independence of observing neighborhood nodes and symmetry in feature space the optimization problem (3) becomes tractable.

IV. EXPERIMENT

In this section, we describe the experimental setup of evaluating the performance of *node2vec* on network classification tasks using common classifiers.

We choose three networks, undirected unweighted graphs, to evaluate the algorithm on:

1. **Cora** [7], a research paper set containing 2,708 machine learning papers which are categorized into 7 classes. The citation relationships between the papers form the edges of the network.
2. **Citeseer** [8], another research paper set containing 3,312 publications and 4,732 connection between them classified into 6 classes
3. **Wiki** [7], a collection of 2,405 web pages with 17,981 links sorted into 19 categories

All three of the chosen data sets represent forms of social networks with the Wiki data set being a much denser network than Cora or Citeseer.

We learn the embeddings on these data sets using *node2vec* for all pairs of the *p* and *q* in [0.1, 0.25, 0.5, 0.75, 1.0]. During extensive preliminary experiments we found that the following hyperparameter values provide relevant representations across used data sets:

- Number of walks per node is set to 100
- Length of walk per node is set to 10
- Representation dimension is set to 128
- Skip-Gram window size is set to 10

We also found that minor changes of those values do not influence the quality of learned representations. The important difference from the hyperparameter values used in the original work is that, unlike them, we used a large number of short walks.

We choose four types of common classifiers to test the embeddings on:

- Ridge regression – a linear regression classifier with L2 regularization
- Nearest centroid classifier – a classification model that assigns labels to unseen data by determining the class of the closest centroid of seen data
- Lasso regression – a linear classifier that models the probability of a class label using L1 regularization
- Support-vector classifier – a non-probabilistic classifier based on support-vector machines with the goal of maximizing the width of the margin while keeping accuracy at a high level

We further examine the performance of the support-vector classifier using both a linear and a *rbf* kernel. For the *rbf* kernel of SV Classifier the gamma hyperparameter was chosen as the inverse of the number of dimensions of feature vectors, which is 128 in our setup.

Since the classification task is a multi-class problem, ridge regression and SVCs were trained using one-vs-rest strategy (OVR). The lasso regression classifier was trained using both one-vs-rest and multinomial multi-class strategies. The nearest centroid classifier is inherently a multi-class classifier.

For all classifiers we performed a 10-fold cross-validation of the regularization parameter for values in [0.01, 0.15, 0.25, 0.5, 2, 4] while using 50% of the data set for that purpose. The model is trained ten times, every time on a different set of 9 folds and validated on the remaining one. The rest of the data set is then used for testing the models.

For each combination of *node2vec* parameters and each classifier we track the accuracy, precision, recall and F1-scores on the test samples.

The accuracy measure is defined as the percentage of correctly classified samples:

$$Accuracy = \frac{TP+TN}{TP+TN+FN+FP} \tag{4}$$

The precision measure is defined as the percentage of positively classified samples which are ground-truth positive:

$$Precision = \frac{TP}{TP+FP} \tag{5}$$

The recall measure is defined as the percentage of correctly classified positive samples:

$$Recall = \frac{TP}{TP+FN} \tag{6}$$

The F1-score is defined as the harmonic mean of precision and recall:

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{7}$$

where TP, FP, TN and FN stand for true positive, false positive, true negative and false negative samples respectively.

Both micro and macro scores were stored for precision, recall and F1-measures. Micro scores are calculated globally by considering all samples from all classes, macro scores are calculated on a per class basis and averaged.

V. RESULTS

In this section, we present and evaluate the results of the experiment on test set. We take the F1-macro score as the main measure of the performance of the classifiers.

We show the *node2vec* parameters as well as the regularization parameters for classifiers of each type that scored the best F1-macro scores for Cora (Table 1), Wiki (Table 2) and Citeseer (Table 3) data sets.

Cora			
Classifier	*p*	*q*	Regularization
Ridge regression	0.75	0.75	0.01
Multinomial lasso regression	0.25	0.25	2.00
Linear SVC	0.75	0.25	2.00
RBF SVC	0.50	0.50	0.15
OVR Lasso regression	0.75	0.25	0.25
Nearest centroid	0.25	0.25	0.15

Table 1 Results on Cora data set for optimal set of parameters

Wiki			
Classifier	*p*	*q*	Regularization
Ridge regression	0.25	1.00	2.00
Multinomial lasso regression	1.00	1.00	0.50
Linear SVC	1.00	0.75	0.50
RBF SVC	0.10	0.50	0.15
OVR Lasso regression	0.10	0.75	0.25
Nearest centroid	0.10	0.75	0.01

Table 2 Results on Wiki data set for optimal set of parameters

Citeseer			
Classifier	*p*	*q*	Regularization
Ridge regression	0.75	0.10	0.50
Multinomial lasso regression	1.00	0.10	0.01
Linear SVC	0.75	0.10	4.00
RBF SVC	0.25	0.10	0.01
OVR Lasso regression	0.75	0.10	0.50
Nearest centroid	0.25	0.10	0.01

Table 3 Results on Citeseer data set for optimal set of parameters

The accuracy and F1-macro scores for the best scoring classifiers of each type on each data set are shown in Table 4 and Table 5 respectively.

Accuracy			
Classifier / Data set	Cora	Wiki	Citeseer
Ridge regression	0.80	0.63	0.57
Multinomial lasso regression	0.82	0.64	0.58
Linear SVC	0.81	0.64	0.58
RBF SVC	0.84	0.67	0.69
OVR Lasso regression	0.82	0.63	0.57
Nearest centroid	0.76	0.57	0.53

Table 4 Accuracy of best performing classifiers across all three data sets

F1-macro			
Classifier / Data set	Cora	Wiki	Citeseer
Ridge regression	0.79	0.44	0.51
Multinomial lasso regression	0.81	0.54	0.52
Linear SVC	0.81	0.54	0.52
RBF SVC	0.83	0.56	0.65
OVR Lasso regression	0.81	0.54	0.52
Nearest centroid	0.74	0.49	0.51

Table 5 F1-macro score of best performing classifiers across all three data sets

As can be seen from measured data, the best performing classifier is the support-vector classifier with an RBF kernel on all data sets. This is not surprising since the RBF SVC is the most complex classifier tested.

The performance difference of the RBF SVC classifier is most notable on Citeseer while on both Wiki and Cora the linear SVC and logistic regression classifiers score similarly. Since RBF SVC is the only non-linear classifier tested, we can conclude that the feature space constructed by *node2vec* on both Cora and Wiki data sets linearly separates the target classes without explicit knowledge of them. The same cannot be said for the Citeseer data set, where the nonlinear nature of RBF SVC gives it a significant advantage.

The classifiers scored high scores across the board on the Cora data set whereas their performance is significantly lower on the Wiki data set. This can be attributed to the larger size and complexity of the Wiki data set.

To further analyze the influence of *node2vec* random walk parameters, we plot the F1-scores of the RBF SVC with respect to *p* and *q* parameters of *node2vec*.

Figures 1 through 3 show the influence of the *q* parameter on the results of the classifier, while Figures 4 through 6 show the same for the *p* parameter. In order to show the influence of a single parameter on classifier performance, we take the average of the scores with respect to the other parameter as the second coordinate of the plotted points.

We can notice easily the difference in quality of the embeddings on the Cora data set compared to Wiki and Citeseer. While varying both *p* and *q* values on Cora leads to the discovery of a local maximum, both Wiki and Citeseer data sets show a tendency towards high *p* values and low *q* values. As high *p* values encourage revisiting of previously seen nodes and high *q* values encourage exploration of deep walks, we can speculate that both the Wiki and the Citeseer data sets show a strong information locality, encoding better embeddings with highly localized walks.

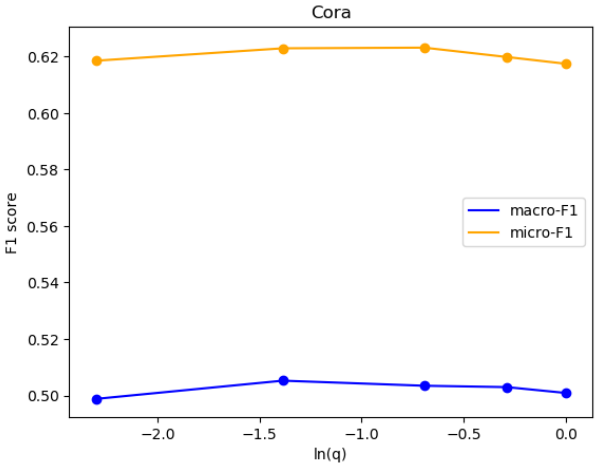

Figure 1 performance of the RBF SVC classifier with relation to *node2vec* parameter *q* on Cora data set

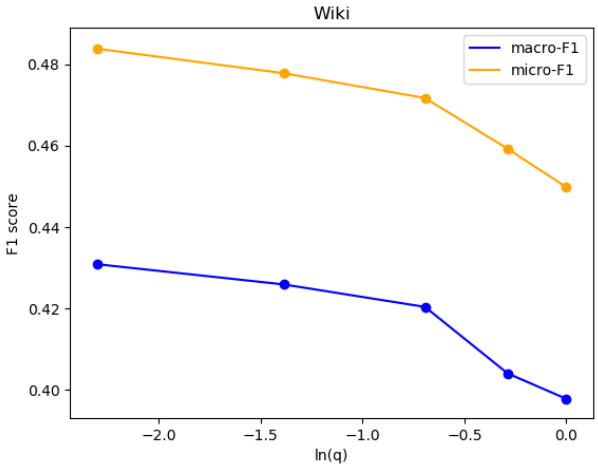

Figure 2 performance of the RBF SVC classifier with relation to *node2vec* parameter *q* on Wiki data set

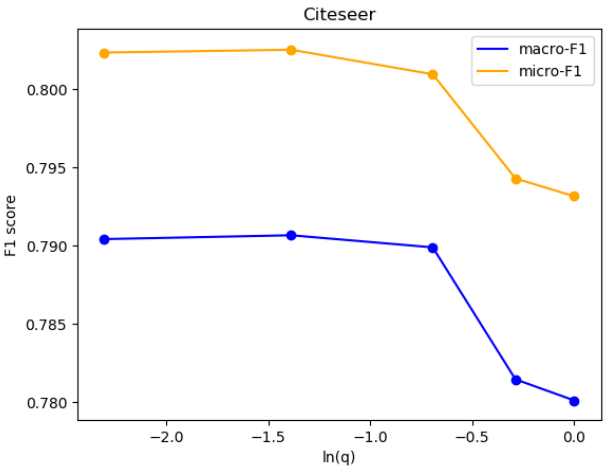

Figure 3 performance of the RBF SVC classifier with relation to *node2vec* parameter *q* on Citeseer data set

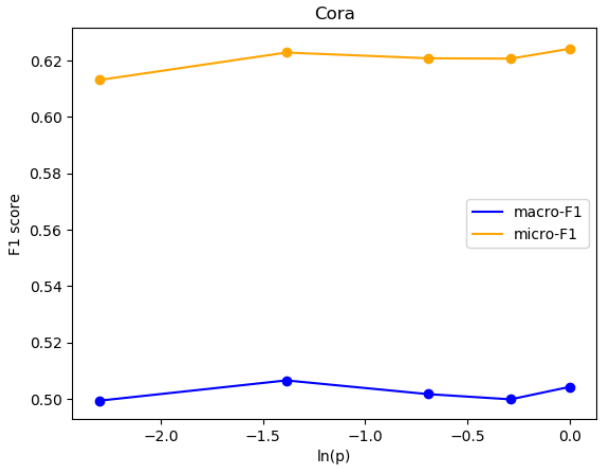

Figure 4 performance of the RBF SVC classifier with relation to *node2vec* parameter *p* on Cora data set

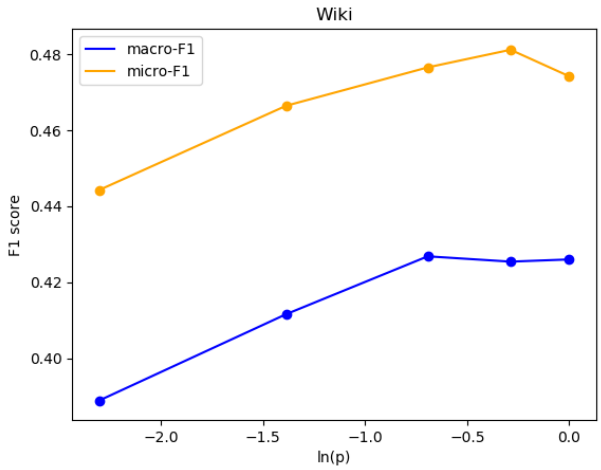

Figure 5 performance of the RBF SVC classifier with relation to *node2vec* parameter *p* on Wiki data set

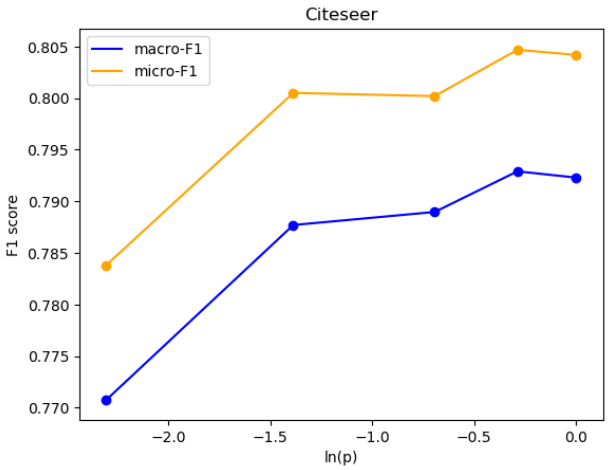

Figure 6 performance of the RBF SVC classifier with relation to *node2vec* parameter *p* on Citeseer data set

VI. Conclusion

In this paper, we studied the performance of common classifiers on network embeddings encoded by *node2vec*. We used four common types of classifiers: ridge classifier, nearest centroid, logistic regression and SVC. We explored the quality of the produced embeddings and the difference between the performance of the different classifiers. We further analyzed the influence of *node2vec* parameters on the results of our best performing classifier – RBF SVC.

To compare the classifiers on a realistic data set, we used freely available social network subset data sets which contain realistic topologies of networks. The used data sets were labeled with multiple classes. We measured the performance of the classifiers using common measures: Precision, Recall and F1-score.

The obtained results were analyzed, and we concluded that the performance of *node2vec* embeddings varies greatly depending on the target network complexity reaching high points of F1-score of 0.83 and low points of 0.56 on the more complex networks. We've also shown how the influence of the random walk parameters of *node2vec* on classifiers' performance can indicate the properties of network relations.

To enforce the given classifiers' results, a detailed statistical analysis could be performed comparing the performance of *node2vec* to a simple baseline approach. Furthermore, an analysis of the influence of all *node2vec* parameters on the classification ability of common classifiers on real-world data sets is left for future work.

Acknowledgments

The authors acknowledge the support of the Croatian Science Foundation for this research through the Reliable Composite Applications Based on Web Services (IP-2018-01-6423) research project. This research has been partly supported by the European Regional Development Fund under the grants KK.01.1.1.01.0009 (DATACROSS) and KK.01.2.1.01.0111 (OperOSS). The Titan X Pascal used for this research was donated by the NVIDIA Corporation.

References

[1] Grover, Aditya, and Jure Leskovec. "node2vec: Scalable feature learning for networks." Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2016.

[2] Representation Learning on Networks, WWW-18 Tutorial [Online], Avaliable: http://snap.stanford.edu/proj/embeddings-www/

[3] L. Tang, and H. Liu, "Relational learning via latent social dimensions," Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009.

[4] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information." Proceedings of the 24th International Joint Conference on Artificial Interlligence, 2015.

[5] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding." Proceedings of the 24th International Conference on World Wide Web, 2015.

[6] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations." Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2014.

[7] A. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning." Information Retrieval Journal, 3, 2000.

[8] P. Sen, G.M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. "Collective classification in network data." AI magazine, 29(3), 2008.

[9] V. Vapnik, S. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation, and signal processing", Mozer M.C., Jordan M.I., and Petsche T. (Eds.) Advances in Neural Information Processing Systems 9, MA, MITPress, Cambridge. pp. 281–287, 1997.

[10] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "DistributedRepresentations of Words and Phrases and their Compositionality," arXiv.org, Oct. 2013.