Supplementary material for the paper: An Empirical Evaluation of Network Representation Learning Methods

1st Alexandru Cristian Mara
Dept. of Electronics and I. S.
Ghent University
Ghent, Belgium
alexandru.mara(at)ugent.be

2nd Jefrey Lijffijt
Dept. of Electronics and I. S.
Ghent University
Ghent, Belgium
jefrey.lijffijt(at)ugent.be

3rd Tijl de Bie
Dept. of Electronics and I. S.
Ghent University
Ghent, Belgium
tijl.debie(at)ugent.be

Hyperparameters and Implementations

In this section we describe the hyperparameters tuned for each embedding method as well as the implementations considered.

DeepWalk: For DeepWalk we evaluate the author's original implementation and the one available in the OpenNE library. In both cases, the Skip-Gram model is approximated via hierarchical softmax. Regarding parameters, we tune the window size, number of walks and walk length.

Node2vec: For Node2vec we also evaluate the original and `OpenNE` implementations both of which approximate the Skip-Gram model via negative sampling. We tune the same parameters as for DeepWalk and additionally the p and q values governing the random walks.

Struc2vec: For struc2vec we only evaluate the original implementation by the authors and tune the same window size, number of walks and walk length as for DeepWalk and Node2vec.

Metapath2vec: In our experiments we consider the original implementation of metapath2vec++ and tune the Skip-Gram learning rate (α) and number of negative samples.

Watch Your Step (WYS): For WYS the original implementation is not directly usable so we resort to evaluating the one provided here. We tune the number of random walks, Skip-Gram window size and learning rate.

Graph Factorization (GF): The implementation of GF we evaluate is the one available in the `OpenNE` library and we did not tune model parameters.

GraRep: For GraRep we study the `OpenNE` implementation and tune the k-step parameter which encodes the order of the transition probability matrix.

HOPE: The `OpenNE` and `GEM` implementations of this method are evaluated and we tune the decay factor β.

Laplacian Eigenmaps (LE): In our empirical evaluation we consider the `OpenNE` and `GEM` implementations of LE. No method-specific hyperparameters are tuned.

Locally Linear Embeddings (LLE): For LLE we evaluate the implementation available in the `GEM` library and no parameters are tuned.

M-NMF: For M-NMF we use the original code by the authors and we tune the number of clusters as hyperparameter.

AROPE: We evaluate the original AROPE code and tune the weight parameter which indicates the order of the proximity. We restrict our analysis to polynomials of order 4.

Structural Deep Network Embedding (SDNE): For SDNE we evaluate the `OpenNE` and `GEM` implementations. We tune the edge reconstruction weight β and the number of neurons and layers of the deep network.

PRUNE: We consider the original implementation of PRUNE and tune the λ parameter controlling the relative importance of node ranking w.r.t. the node proximity in the graph.

VERSE: In our experiments we evaluate the Python implementation released by the authors and tune the number of negative samples ($nsamples$).

GAE/VGAE: We evaluate the original implementations released by the authors for both models and do not tune any hyperparameters.

GIN/GatedGCN: For both approaches we adapt the code available in the following repository. Our modified version is made available here. As hyperparameters we tune the learning rate and the maximum number of iterations each approach is trained for.

LINE: For LINE we evaluate the original and `OpenNE` implementations and tune ρ, the initial learning rate and *negative_ratio*, the number of non-edges per true edge used in the optimization procedure.

CNE: For CNE only the original implementation is available and we tune the learning rate parameter.

Performance variations

In this section we present a number of software and method implementation-related issues encountered during the benchmark process. Specifically, for Metapath2vec we have found an important degradation in performance when parallelism is increased beyond 4 threads. Table I summarizes the average AUROC, standard deviation and p-values over 5 independent runs of the method using 4 and 8 thread parallelism. The results are presented for all edge embedding heuristics considered in

TABLE I
MEAN AND STD DEVIATION OF AUC-ROC FOR LINE AND METAPATH2VEC WITH 4 AND 8 THREADS. RESULTS ARE ONLY SHOWN FOR THE FACEBOOK, PPI AND ASTROPH NETWORKS. THE LAST COLUMN REPORTS P-VALUES COMPUTED FROM 5 INDEPENDENT RUNS WITH 4 AND 8 THREADS.

Dataset	Method	4 threads	8 threads	p
Facebook	LINE(avg.)	0.731 ± 0.005	0.002	0.283
	LINE(had.)	0.968 ± 0.001	0.964 ± 0.002	0.007
	LINE(W L_1)	0.682 ± 0.016	0.679 ± 0.013	0.707
	LINE(W L_2)	0.69 ± 0.017	0.686 ± 0.014	0.703
	Metapath2vec(avg.)	0.75 ± 0.003	0.73 ± 0.003	$4.30 \cdot 10^{-6}$
	Metapath2vec(had.)	0.89 ± 0.007	0.857 ± 0.006	$5.45 \cdot 10^{-5}$
	Metapath2vec(W L_1)	0.912 ± 0.005	0.912 ± 0.008	0.920
	Metapath2vec(W L_2)	0.896 ± 0.005	0.889 ± 0.009	0.205
PPI	LINE(avg.)	0.824 ± 0.001	0.821 ± 0.003	0.074
	LINE(had.)	0.763 ± 0.007	0.756 ± 0.008	0.189
	LINE(W L_1)	0.763 ± 0.003	0.761 ± 0.002	0.319
	LINE(W L_2)	0.766 ± 0.003	0.764 ± 0.002	0.364
	Metapath2vec(avg.)	0.839 ± 0.002	0.837 ± 0.006	0.661
	Metapath2vec(had.)	0.824 ± 0.003	0.822 ± 0.005	0.416
	Metapath2vec(W L_1)	0.706 ± 0.009	0.702 ± 0.005	0.422
	Metapath2vec(W L_2)	0.692 ± 0.006	0.69 ± 0.008	0.686
AstroPH	LINE(avg.)	0.869 ± 0.001	0.869 ± 0.001	0.984
	LINE(had.)	0.982 ± 0.001	0.982 ± 0.001	0.878
	LINE(W L_1)	0.628 ± 0.012	0.624 ± 0.01	0.512
	LINE(W L_2)	0.636 ± 0.012	0.631 ± 0.009	0.463
	Metapath2vec(avg.)	0.768 ± 0.002	0.77 ± 0.005	0.360
	Metapath2vec(had.)	0.828 ± 0.006	0.837 ± 0.007	0.054
	Metapath2vec(W L_1)	0.646 ± 0.006	0.641 ± 0.01	0.336
	Metapath2vec(W L_2)	0.642 ± 0.006	0.638 ± 0.01	0.419

our evaluation using experimental setup LP2. The original implementation of LINE is shown for reference. The results using 1 and 16 threads, omitted in the table, are very similar to those obtained with 4 and 8 threads respectively. Our results indicate statistically significant differences at the 0.01 level for Metapath2vec on the Facebook dataset for different levels of parallelism.

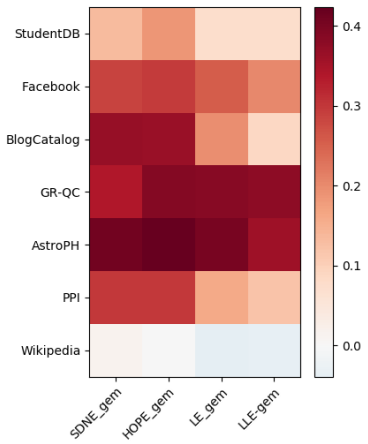

Fig. 1. Difference in LP AUC-ROC for two sets of GEM dependencies. Both sets satisfy the loose version requirements of the library, yet provide very different results.

Throughout our experiments we have also found strong variations in performance for all methods in the GEM library. These fluctuations in AUROC of up to 44.5% appear to be related to the versions of GEM dependencies installed. The authors of GEM do provide a list of minimal required versions for these libraries, however, there are many possible combinations which satisfy these premisses. In Figure 1 we present the difference un AUROC for two such sets of GEM dependencies. The average AUROC over all methods and datasets in one case is 0.673 while in the second is 0.902. In the figure, we observe that most of this variation is concentrated around the AstroPH, GR-QC and BlogCatalog datasets. The results of SDNE and HOPE also fluctuate more than those of LE and LLE.