EvalNE: A Framework for Evaluating Network Embeddings on Link Prediction

Alexandru Mara, Jefrey Lijffijt, and Tijl De Bie

Research Interest

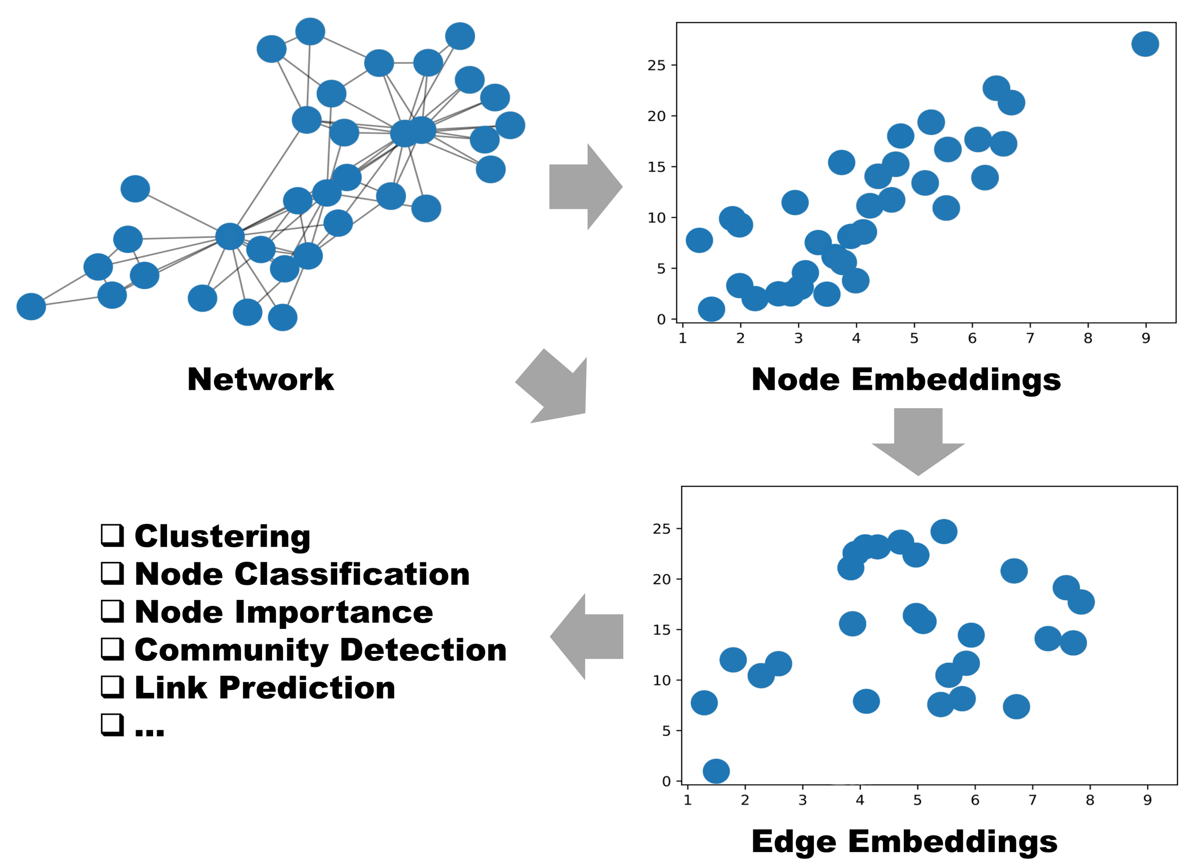
- Address the **reproducibility crisis** in the field of Network Embeddings (NE) for Link Prediction (LP).
- **Facilitate evaluation** of NE methods and comparison to existing ones.

Motivation

- Non-standard evaluation ➡ incomparable methods
- LP a complex task ➡ evaluation errors
- Current libraries ➡ limited baselines

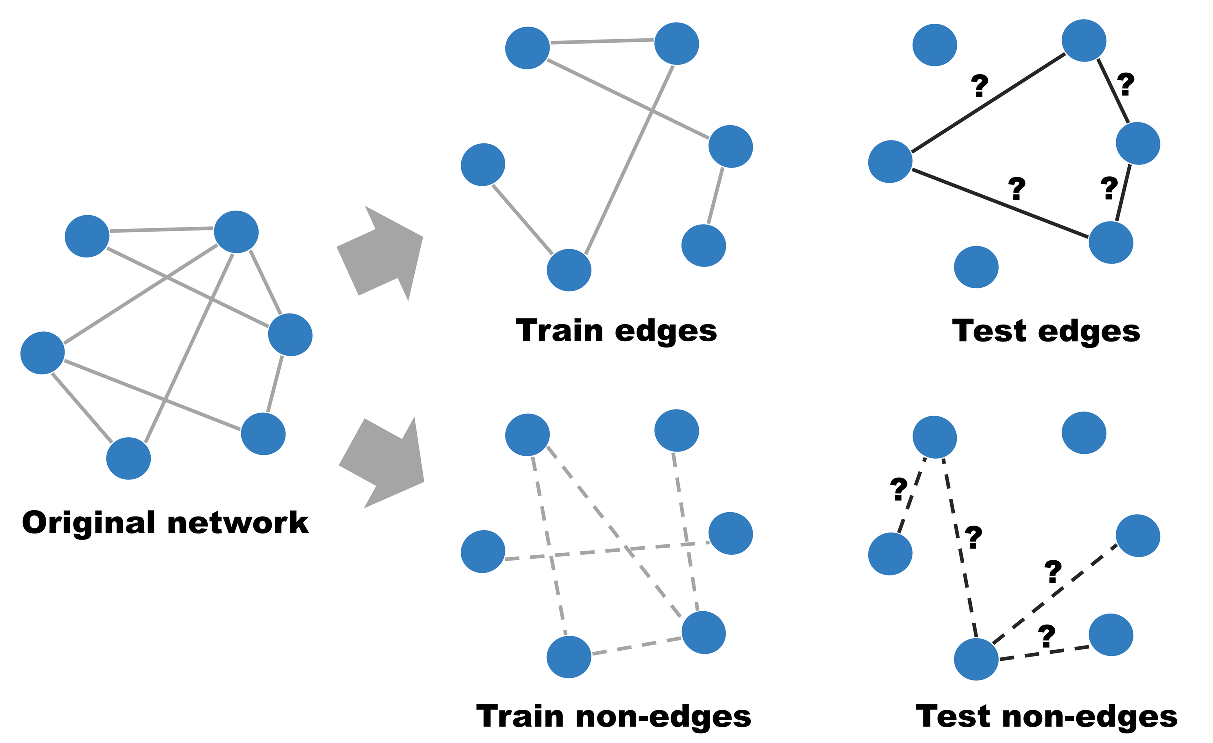
Network Embeddings

- A mapping of network nodes to *d*-dimensional vector representations
- Node embeddings and edge embeddings
- After embedding a network, standard machine learning tasks can be performed (e.g. clustering, link prediction)



Link Prediction

- Estimate the likelihood of the existance of edges. between pairs of nodes.
- Both true edges and non-edges required for evaluation.

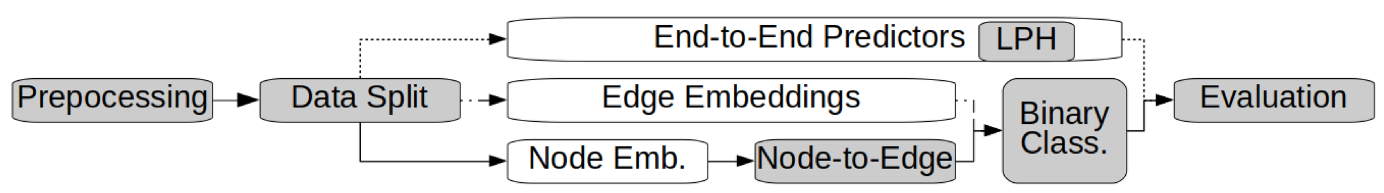


Evaluation choices

- Network Preprocessing
- Train/test fraction
- Edge splits
- Non-Edge sampling
- Node to Edge embeddings
- LP heuristics
- Binary classifiers
- Evaluation metrics

Our Framework

- Command line tool and API.
- Easily replicate any evaluation setup.
- Automate the method evaluation process.
- Run NE methods code in any language



Toolbox usage

1. Fill the conf file
2. Run: $ python evalne conf.ini

```
[GENERAL]
EDGE_EMBEDDING_METHODS = average hadamard
LP_MODEL = LogisticRegression
EXP_REPEATS = 10
EMBED_DIM = 128
VERBOSE = True

[NETWORKS]
NAMES = Facebook PPI ArXiv
INPATHS = ../data/Facebook/facebook_combined.txt
          ../data/PPI/ppi.edgelist
          ../data/Astro-PH/CA-AstroPh.txt
OUTPATHS = ../output/Facebook/
           ../output/PPI/
           ../output/Astro-Ph/
DIRECTED = False False False
SEPARATORS = '\s' ',' '\t'
COMMENTS = '#' '#' ';'

[BASELINES]
LP_BASELINES = common_neighbours
               jaccard_coefficient
               adamic_adar_index
               preferential_attachment
NEIGHBOURHOOD = in out

[OPENNE METHODS]
NAMES_OPNE = node2vec deepWalk line
METHODS_OPNE = python -m openne --method node2vec --epochs 100
               python -m openne --method deepWalk --epochs 100
               python -m openne --method line --epochs 100
TUNE_PARAMS_OPNE = --p 0.25 0.5 1 2 4 --q 0.25 0.5 1 2 4

[OTHER METHODS]
NAMES_OTHER = prune
EMBTYPE_OTHER = ne
METHODS_OTHER = python ../methods/PRUNE/src/main.py --inputgraph {} --output {} --dimension {}
#                ../methods/metapath2vec/metapath2vec -train {} -output {} -size {}
TUNE_PARAMS_OTHER = -negative 1 5 10
INPUT_DELIM_OTHER = '\s'
OUTPUT_DELIM_OTHER = ','
```

```
[PREPROCESSING]
RELABEL = True
DEL_SELFLOOPS = True
PREP_NW_NAME = prep_graph.edgelist
WRITE_STATS = True
DELIMITER = ','

[TRAINTEST]
TRAIN_FRAC = 0.5
FAST_SPLIT = True
OWA = True
NUM_FE_TRAIN = None
NUM_FE_TEST = None
TRAINTEST_PATH = train_test_splits/

[REPORT]
MAXIMIZE = auroc
SCORES = %(maximize)s
CURVES = roc
PRECATK_VALS = 2 10 100 200 500 800 1000
```

Experimental Results

- Reproducing Node2vec [1] experiments:

	Facebook	PPI	arXiv	Sum of diffs
Common Neighbors	0,1691	0,0617	0,1377	0,3685
Jaccard's Coefficient	0,0871	0,0626	0,1461	0,2958
Adamic-Adar	0,1517	0,0661	0,122	0,3398
Pref. Attachment	0,1248	0,237	0,1745	0,5363
DeepWalk	0,0173	0,0799	0,0139	0,1111
LINE	0,0417	0,1618	0,0805	0,284
node2vec	0,0249	0,0248	0,0106	0,0603
DeepWalk	0,0301	0,2035	0,1358	0,3694
LINE	0,0956	0,1079	0,0472	0,2507
node2vec	0,0216	0,2188	0,0638	0,3042
DeepWalk	0,0081	0,1414	0,09	0,2395
LINE	0,051	0,1665	0,1183	0,3358
node2vec	0,0133	0,052	0,0602	0,1255
DeepWalk	0,0066	0,1339	0,0906	0,2311
LINE	0,0747	0,1773	0,1521	0,4041
node2vec	0,0122	0,0505	0,0598	0,1225
Sum of diffs	0,9298	1,9457	1,5031	

- Reproducing CNE [2] experiments:

	Facebook	PPI	arXiv	BlogCatalog	Wikipedia	studentdb	Sum of diffs
Common Neighbor	0,0056	0,0066	0,0108	0,0039	0,0101	0,0028	0,04
Jaccard Sim.	0,0046	0,0064	0,0106	0,0037	0,0104	0,0028	0,04
Adamic Adar	0,0055	0,0068	0,0108	0,0041	0,0086	0,0028	0,04
Prefere. Attach.	0,0090	0,0148	0,0101	0,0028	0,0050	0,0122	0,05
Deepwalk	0,0441	0,0955	0,1057	0,1026	0,0767	0,0598	0,48
LINE	0,0246	0,1402	0,0726	0,0562	0,1700	0,0760	0,54
node2vec	0,0548	0,1194	0,1038	0,2043	0,1843	0,1575	0,82
metapath2vec++	0,0454	0,1765	0,1620	0,0402	0,0190	0,1339	0,58
CNE(uniform)	0,0017	0,0045	0,0003	0,0082	0,0010	0,0023	0,02
CNE(degree)	0,0028	0,0000	0,0000	0,0001	0,0067	0,0043	0,01
Sum of diffs	0,1981	0,5707	0,4867	0,4261	0,4918	0,4544	

- Reproducing PRUNE [3] experiments:

	Hep-Ph	Webspam	Sum of diffs
DeepWalk	0,1395	0,1334	0,2729
LINE	0,1243	0,223	0,3473
node2vec	0,1502	0,203	0,3532
SDNE	0,0275	0,2017	0,2292
PRUNE	0,108	0,1814	0,2894
Sum of diffs	0,5495	0,9425	

- Scalability experiments:

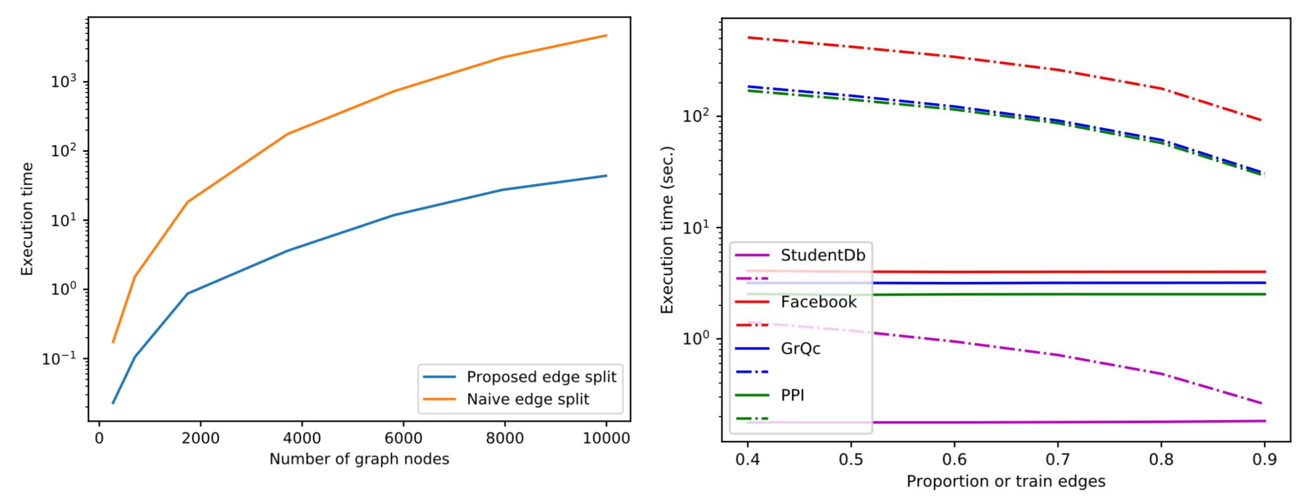


Figure 1: Scalability plots showing the evolution of the execution time w.r.t. a) the number of node in a graph and b) the proportion of train and test edges requested.

Acknowledgements:

This work was supported by the ERC under the EU's Seventh Framework Programme (FP7/2007-2013) / ERC Grant Agr. no. 615517, FWO (project no. G091017N, G0F9816N), the EU's Horizon 2020 research and innovation programme and the FWO under the Marie Sklodowska-Curie Grant Agr. no. 665501.

References:

[1] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In Proc. of the 22nd ACM SIGKDD (KDD '16). New York, USA, 855-864.
[2] Bo Kang, Jefrey Lijffijt and Tijl De Bie. 2019. Conditional Network Embeddings. To appear in Proc. of ICLR.
[3] Yi-An Lai, Chin-Chi Hsu, Wen Hao Chen, Mi-Yen Yeh and Shou-De Lin. 2017. PRUNE: Preserving Proximity and Global Ranking for Network Embedding. In Proceedings of NIPS 2017, 5257-5266.

Contact:

alexandru.mara@UGent.be