

Seminar II: Deep Learning-based Natural Language Processing

**“node2vec: Scalable Feature Learning for Networks”
Paper review**

Contents

1. Introduction

- Graphs
- Graph Representation
- Homophily & Structural Equivalence
- BFS & DFS
- Problem statement
- Key Contribution

2. Methodology

- Feature learning framework
- Biased random walk
- The node2vec algorithm
- Learning edge features

3. Results and Discussion

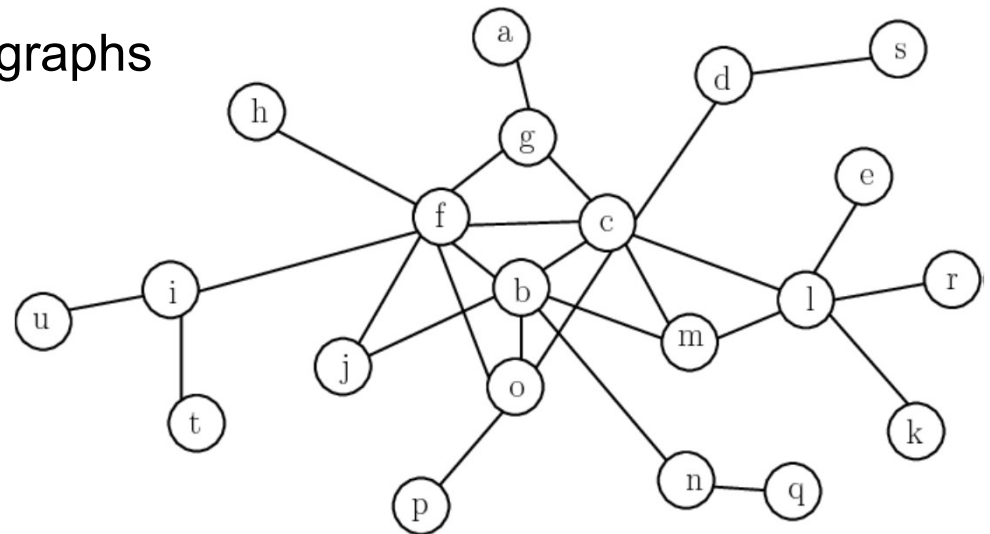
4. Conclusions

Introduction. Graphs

Data consist of

- Vertices (Nodes) V
- Edges (Connections) E
 - Directed or Undirected
 - Weights

Words and images are a special case of graphs

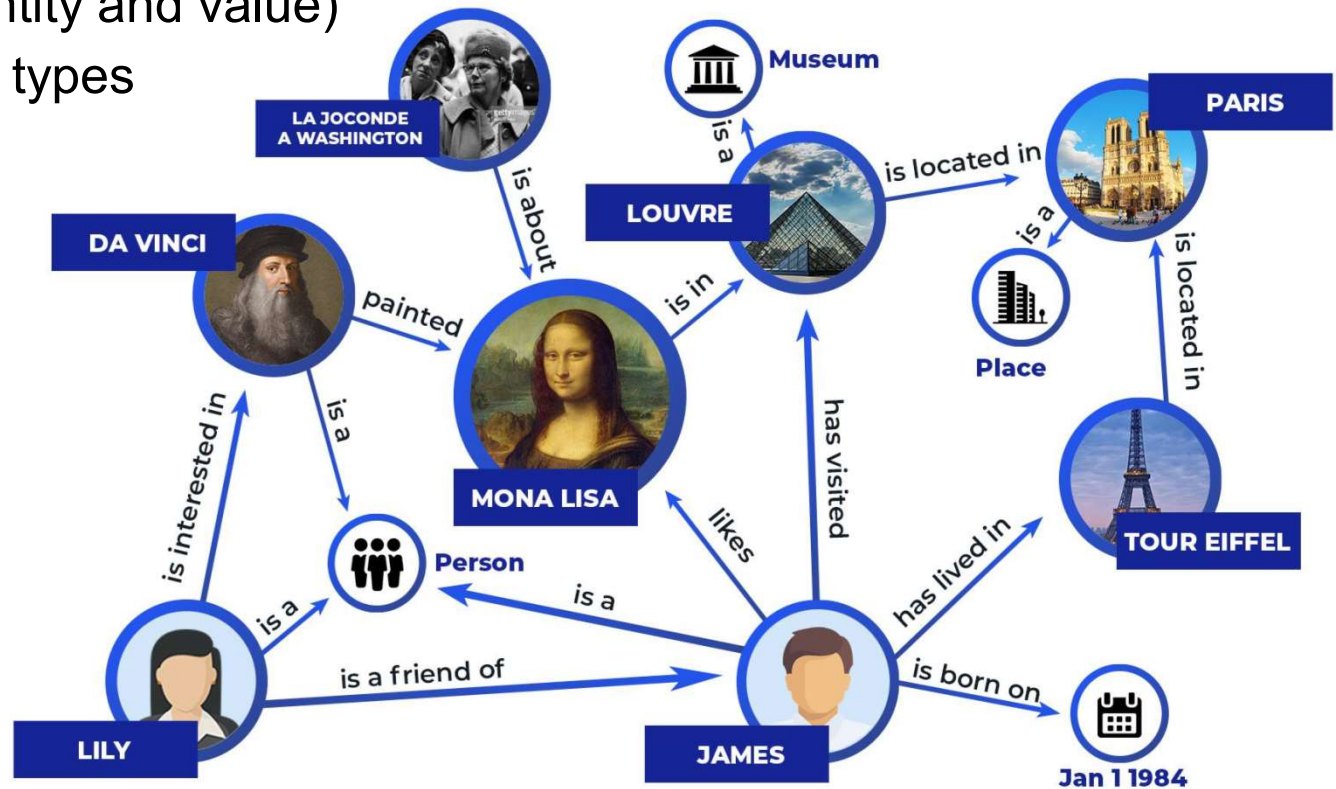


Introduction. Graphs

Everything is a graph

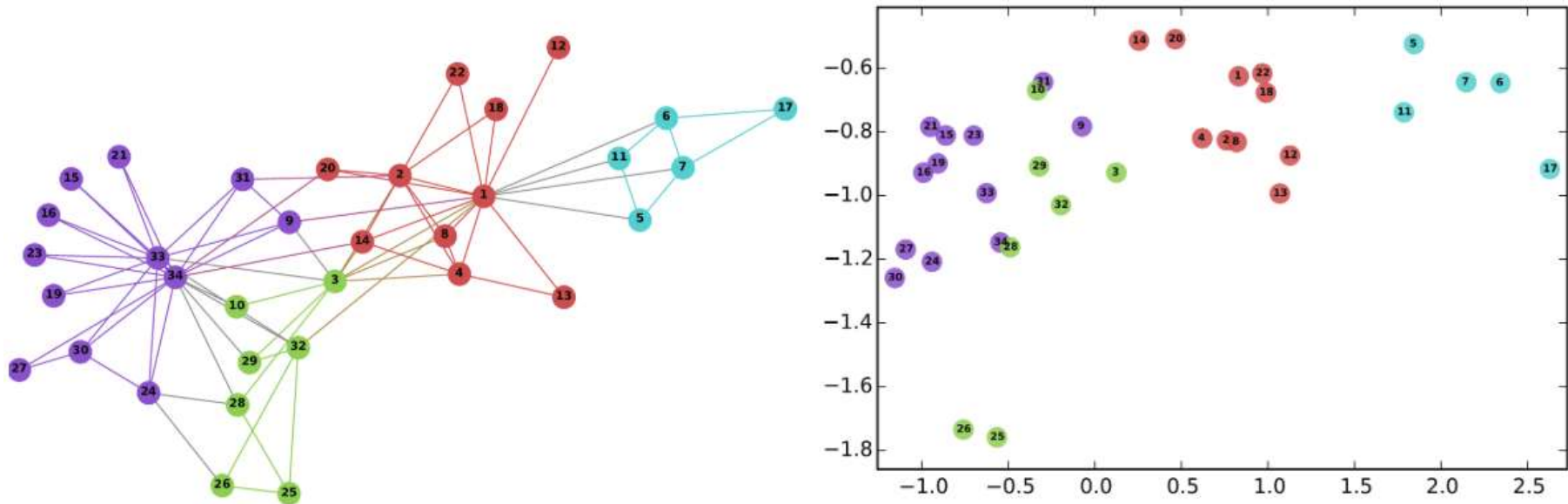
Example:

- Multiple node types (entity and value)
- Directed Multiple edge types



Introduction. Graphs Representation

Most of the time: node representation



Source: Google AI blog

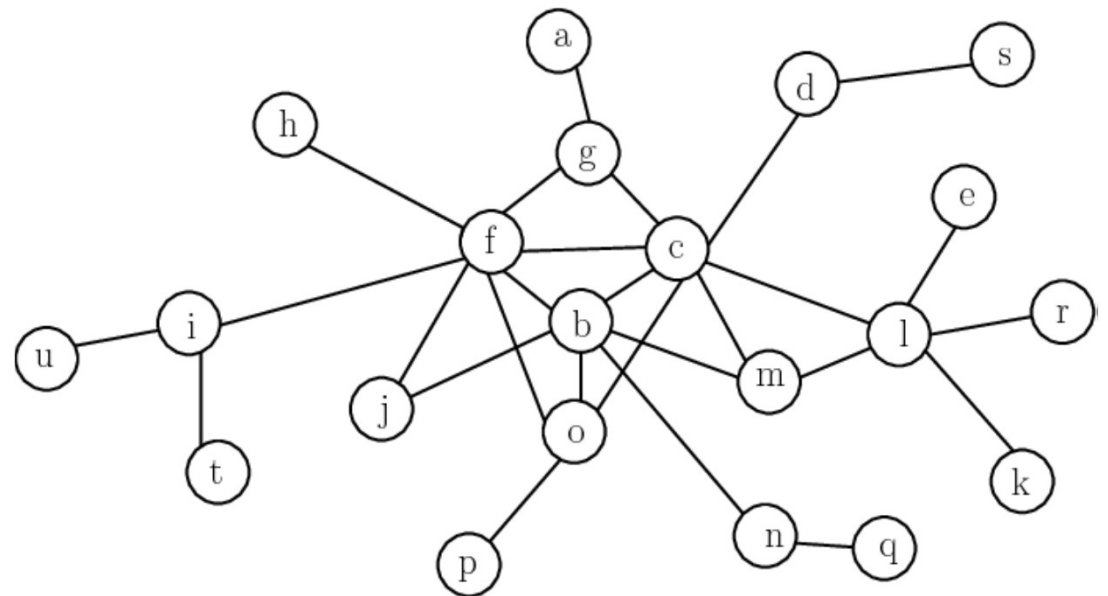
Left: The well-known [Karate](#) graph representing a social network.

Right: A continuous space embedding of the nodes in the graph using [DeepWalk](#).

Introduction. Graphs Representation

The problem is graphs are not linear.

If we could sample multiple random linear sequence from graphs, then we could pass them to word2vec algorithm (etc. skip-gram).



Introduction. Homophily & Structural Equivalence

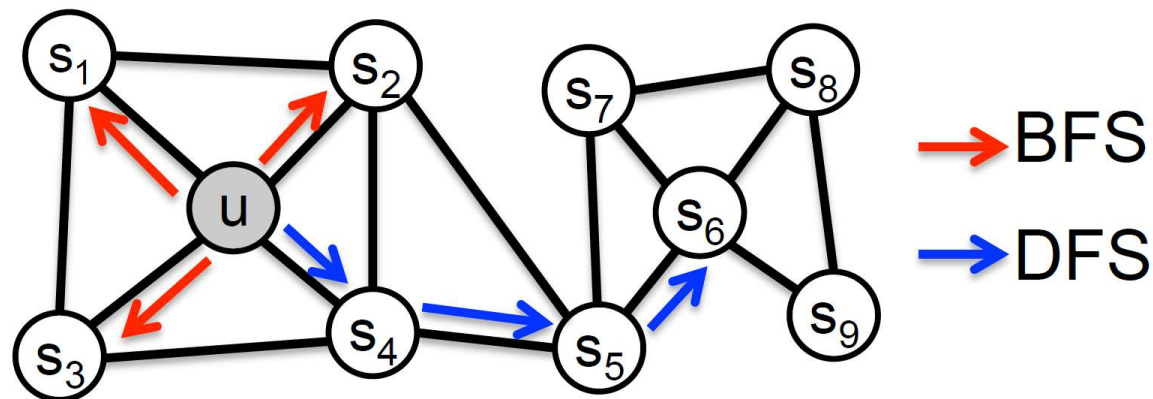
What kind of representation do we want?

Homophily: similar network clusters (ex: s_1 and u)

Structural equivalence: similar structural roles (ex: s_6 and u)

BFS closely lead to structural embeddings

DFS closely lead to homophily



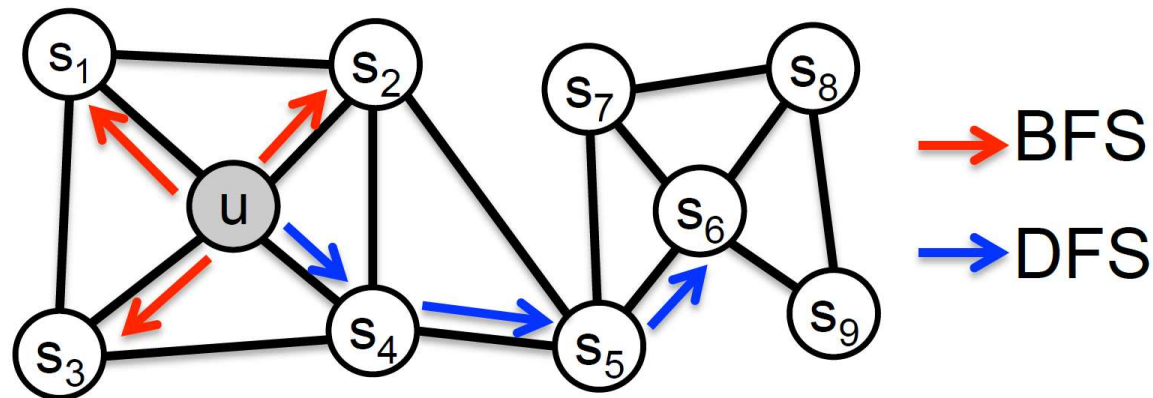
*BFS and DFS search strategies from node u ($k = 3$)
Source: node2vec: Scalable Feature Learning for Networks*

Introduction. BFS & DFS

Suppose k is the size of neighborhood = 3

BFS samples: s_1 s_2 s_3

DFS samples: s_4 s_5 s_6



*BFS and DFS search strategies from node u ($k = 3$)
Source: node2vec: Scalable Feature Learning for Networks*

Introduction. Problem statement

PROBLEMS

- Unsupervised feature learning (e.g., PCA) suffer from computational costs because eigen decomposition is expensive
- If we turn a network into a ordered sequence of nodes, we can pass it to word2vec
- There are many possible sampling strategies
- The MAJOR GAP is that there is no flexibility in sampling nodes from a network

Introduction. Key Contribution

OBJECTIVE

To design a flexible sampling technique that is not tied to a particular sampling strategy and provides hyper-parameters to tune the explored search space.

Methodology - Feature learning framework

Let $G = (V, E)$

Let $f: V \rightarrow \mathbb{R}^d$

Every $u \in V$, $N_s(u) \subset V$ as a network neighborhood of node u generated through a neighborhood sampling strategy S

Extending the Skip-gram architecture to networks

$$\max_f \sum_{u \in V} \log \Pr(N_s(u) | f(u))$$

Networks are not linear, and thus richer notion of a neighborhood is needed

The neighborhoods are not restricted just immediate neighbors but can have vastly different structures depending on the sampling strategy S

Simply, how do we decide which nodes are neighbors given u

Methodology - Biased random walk

2nd order random walk with two hyper-parameters p and q

2nd order means that it considers the previous state where it came from

Unnormalized transition probability:

$$\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx} \text{ where}$$

$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

Normalized transition probability:

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

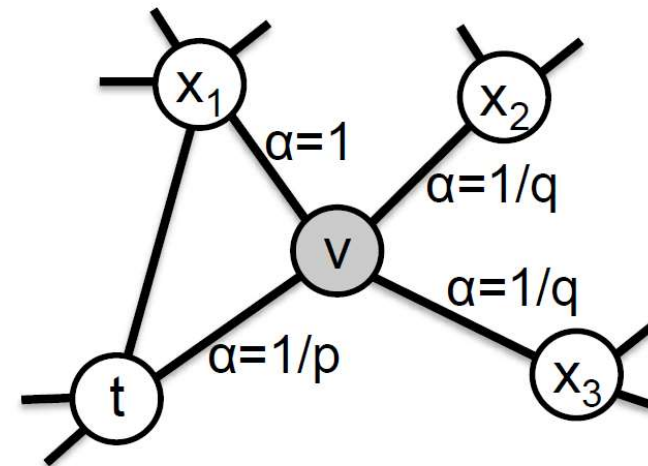


Illustration of the random walk procedure in node2vec. The walk just transitioned from t to v and is now evaluating its next step out of node v. Edge labels indicate search biases

Source: node2vec: Scalable Feature Learning for Networks

Methodology - The node2vec algorithm

Algorithm 1 The *node2vec* algorithm.

LearnFeatures (Graph $G = (V, E, W)$, Dimensions d , Walks per node r , Walk length l , Context size k , Return p , In-out q)
 $\pi = \text{PreprocessModifiedWeights}(G, p, q)$
 $G' = (V, E, \pi)$
 Initialize *walks* to Empty
 for $iter = 1$ **to** r **do**
 for all nodes $u \in V$ **do**
 $walk = \text{node2vecWalk}(G', u, l)$
 Append $walk$ to *walks*
 $f = \text{StochasticGradientDescent}(k, d, \text{walks})$
 return f

node2vecWalk (Graph $G' = (V, E, \pi)$, Start node u , Length l)
 Initialize *walk* to $[u]$
 for $walk_iter = 1$ **to** l **do**
 $curr = walk[-1]$
 $V_{curr} = \text{GetNeighbors}(curr, G')$
 $s = \text{AliasSample}(V_{curr}, \pi)$
 Append s to *walk*
 return *walk*

Methodology - Learning edge features

Given two nodes u and v

We can define $g(u, v) = f(u) \circ f(v)$

Such that $g: V \times V \rightarrow \mathbb{R}^d$

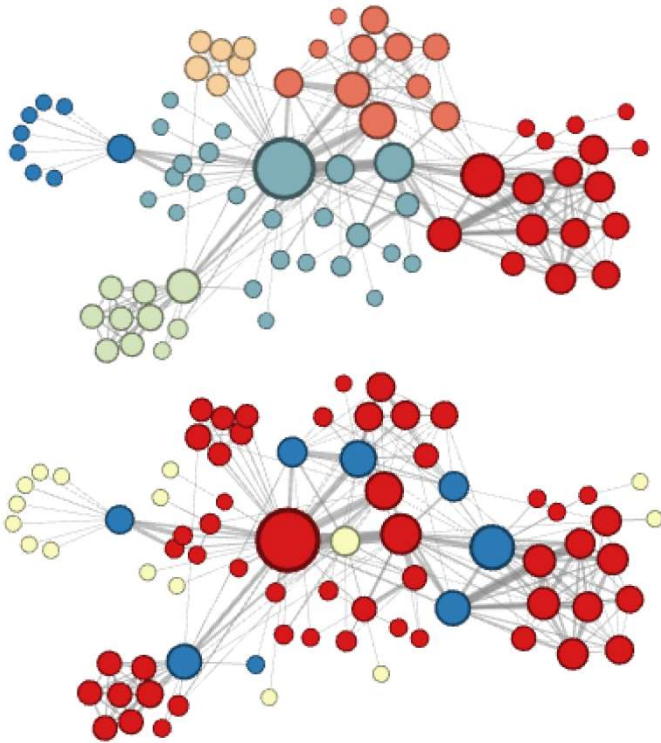
Choice of binary operators \circ could be

- Average
- Hadamard
- L1 norm
- L2 norm
- ...

For example, we could predict whether a link exist between two nodes

Binary classification task

Results and Discussions

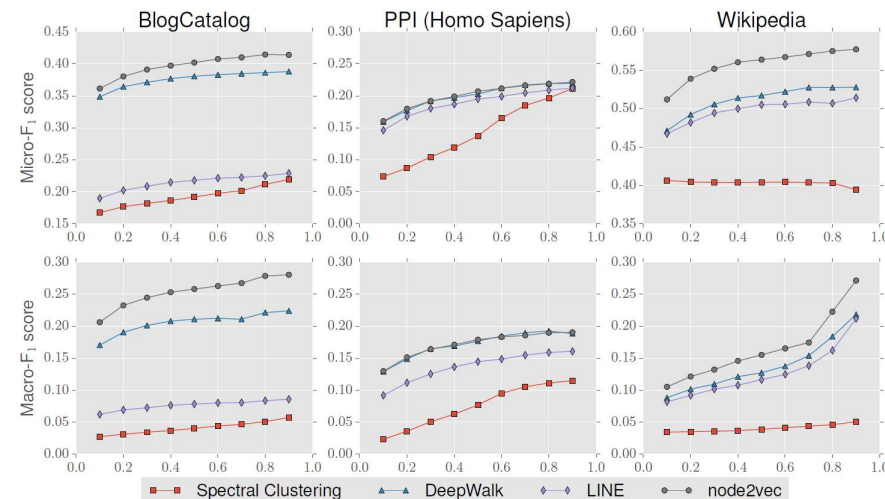


Complementary visualizations of Les Misérables coappearance network generated by node2vec with label colours reflecting homophily (top) and structural equivalence (bottom).

Source: node2vec: Scalable Feature Learning for Networks

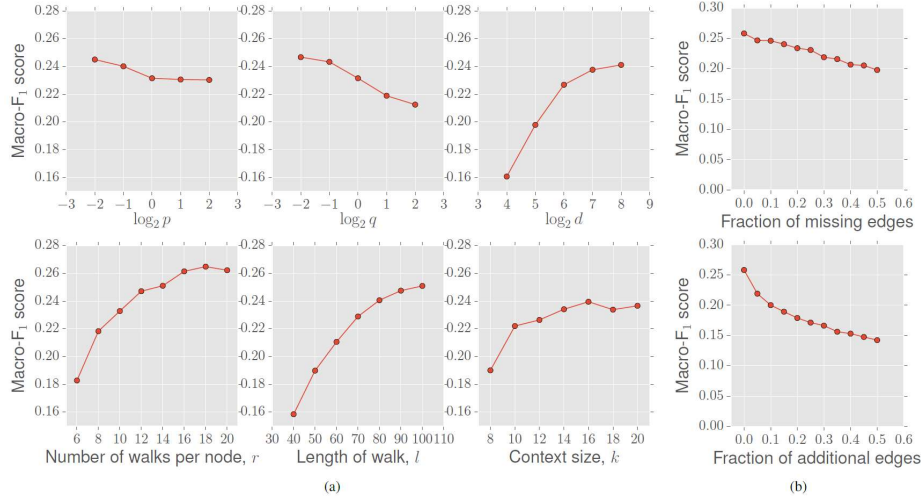
Algorithm	Dataset		
	BlogCatalog	PPI	Wikipedia
Spectral Clustering	0.0405	0.0681	0.0395
DeepWalk	0.2110	0.1768	0.1274
LINE	0.0784	0.1447	0.1164
node2vec	0.2581	0.1791	0.1552
node2vec settings (p,q)	0.25, 0.25	4, 1	4, 0.5
Gain of node2vec [%]	22.3	1.3	21.8

Macro-F1 scores for multilabel classification on BlogCatalog, PPI (Homo sapiens) and Wikipedia word cooccurrence networks with 50% of the nodes labelled for training.

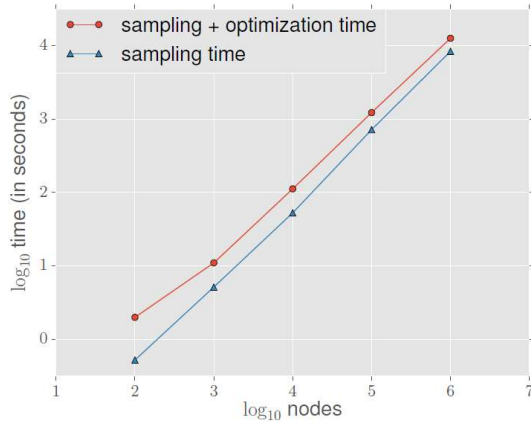


Performance evaluation of different benchmarks on varying the amount of labelled data used for training. The x axis denotes the fraction of labelled data

Results and Discussions



(a). Parameter sensitivity (b). Perturbation analysis for multilabel classification on the BlogCatalog network.



Scalability of node2vec on Erdos-Renyi graphs with an average degree of 10.

Op	Algorithm	Dataset		
		Facebook	PPI	arXiv
(a)	Common Neighbors	0.8100	0.7142	0.8153
	Jaccard's Coefficient	0.8880	0.7018	0.8067
	Adamic-Adar	0.8289	0.7126	0.8315
	Pref. Attachment	0.7137	0.6670	0.6996
	Spectral Clustering	0.5960	0.6588	0.5812
(b)	DeepWalk	0.7238	0.6923	0.7066
	LINE	0.7029	0.6330	0.6516
	node2vec	0.7266	0.7543	0.7221
	Spectral Clustering	0.6192	0.4920	0.5740
(c)	DeepWalk	0.9680	0.7441	0.9340
	LINE	0.9490	0.7249	0.8902
	node2vec	0.9680	0.7719	0.9366
	Spectral Clustering	0.7200	0.6356	0.7099
(d)	DeepWalk	0.9574	0.6026	0.8282
	LINE	0.9483	0.7024	0.8809
	node2vec	0.9602	0.6292	0.8468
	Spectral Clustering	0.7107	0.6026	0.6765
(e)	DeepWalk	0.9584	0.6118	0.8305
	LINE	0.9460	0.7106	0.8862
	node2vec	0.9606	0.6236	0.8477
	Spectral Clustering	0.7107	0.6026	0.6765

Area Under Curve (AUC) scores for link prediction. Comparison with popular baselines and embedding based methods bootstrapped using binary operators: (a) Average, (b) Hadamard, (c) Weighted-L1, and (d) Weighted-L2

Conclusions

- BFS suitable for structural equivalences
- DFS suitable for homophiles clusters
- node2vec is both flexible and controllable exploring neighborhoods through parameters p and q
- node2vec is scalable and robust to perturbations

Future work:

- To explore the reason why Hadamard operator is success in the link prediction task