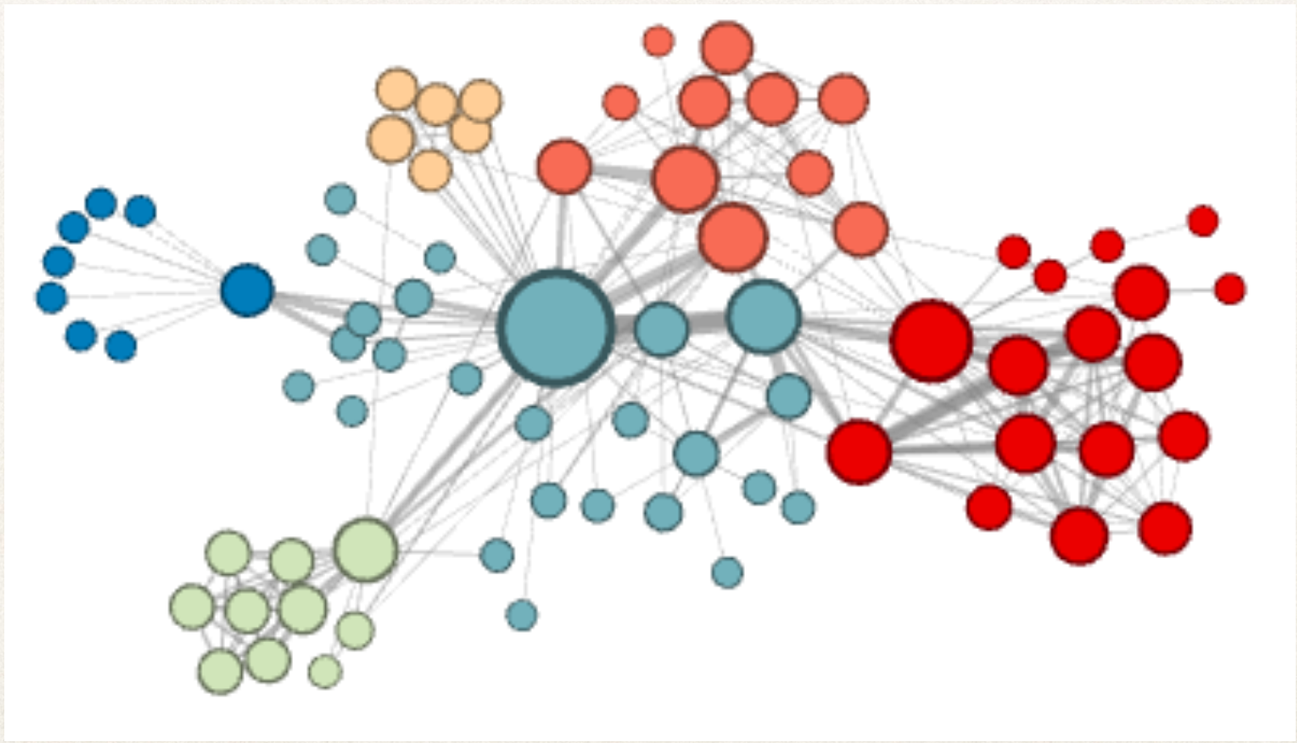
node2vec

Scalable Feature Learning for Networks (Aditya Grover, Jure Leskovec - Stanford)

TADA S2 : Network Analysis

Abtract

- Algorithmic framework for learning continuous feature representations for nodes in networks.
  - Captures the diversity of connectivity patterns observed in networks.
  - One of the early works (2016) in a domain of machine learning with graphs
- **Mapping of nodes to a low-dimensional space** of features that maximizes the likelihood of preserving network neighborhoods of nodes.



Introduction

- Semi-supervised algorithm for scalable feature learning in networks.
- Optimized a custom graph-based objective function using SGD motivated by prior work on natural language processing (skip-gram).
- In order to maximize the likelihood of preserving network neighborhoods of nodes in a d-dimensional feature space - a 2nd order random walk approach is used to generate (sample) network neighborhoods for nodes.
- Similar to Deep Walk - generates embeddings from the network using random walks (+ adds controllable factor in generating random walks)

Bottleneck

- Problem :
  - It's difficult to take a network's raw representation and use it to build a model
- Solution :
  - Representation of each node's features are used to train a model
  - Previous works
    - Manually creating features based on the degree (PageRank)
    - Learning feature based on matrix factorization - do not scale well to real-world networks
    - Deep learning methods - high cost training time

Current Work

- Learn mapping f so that map's any node can be projected as a **vector in d dimensional subspace.**
  - Embedding nodes with similar network neighbors close in the feature space
  - via maximum likelihood optimization
  - objective function that does not restrict the notion of neighborhoods - second-order random walk procedures
  - random walks : space and time complexity more efficient

Flexible notion of neighborhoods

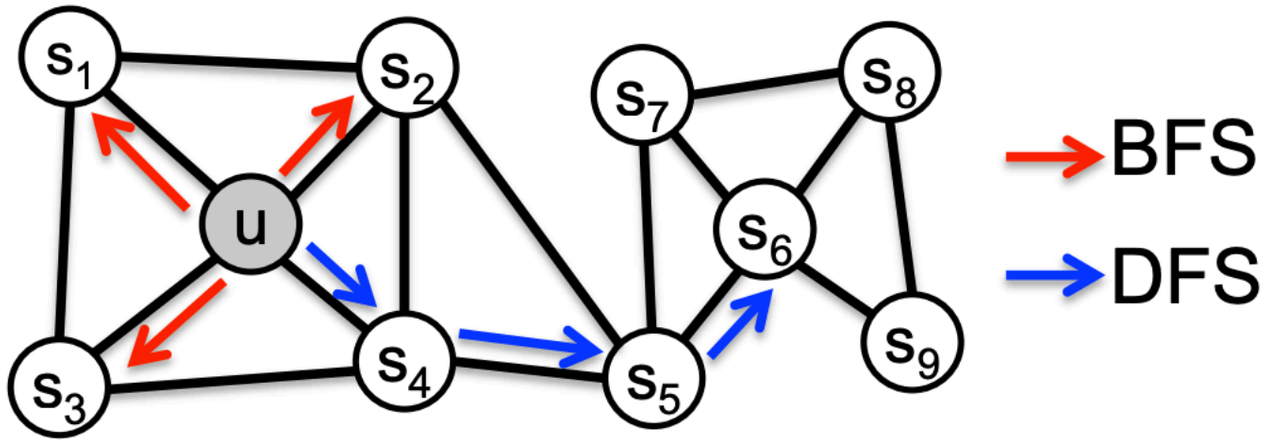


Figure 1: BFS and DFS search strategies from node u ($k = 3$).

- Interpolates between BFS (local, homophily), DFS (high global, structural equivalence) to gain both microscopic and macroscopic view of the network.
  - "For example, we expect the social network of bloggers to exhibit strong homophily-based relationships; however, there might also be some “familiar strangers”, i.e., bloggers that do not interact but share interests and hence are structurally equivalent nodes.
  - The biological states of proteins in a protein-protein interaction network also exhibit both types of equivalences. For example, they exhibit structural equivalence when proteins perform functions complementary to those of neighboring proteins, and at other times, they organize based on homophily in assisting neighboring proteins in performing similar functions"

Flexible notion of neighborhoods

3.2.1 *Random Walks*

Formally, given a source node u, we simulate a random walk of fixed length l. Let c_i denote the ith node in the walk, starting with $c_0 = u$. Nodes c_i are generated by the following distribution:

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

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

3.2.2 *Search bias α*

We define a 2nd order random walk with two parameters p and q which guide the walk: Consider a random walk that just traversed edge (t, v) and now resides at node v (Figure 2). The walk now needs to decide on the next step so it evaluates the transition probabilities π_{vx} on edges (v, x) leading from v. We set the unnormalized transition probability to $\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$, where

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

and d_{tx} denotes the shortest path distance between nodes t and x. Note that d_{tx} must be one of $\{0, 1, 2\}$, and hence, the two parameters are necessary and sufficient to guide the walk.

Intuitively, parameters p and q control how fast the walk explores and leaves the neighborhood of starting node u. In particular, the parameters allow our search procedure to (approximately) interpolate between BFS and DFS and thereby reflect an affinity for different notions of node equivalences.

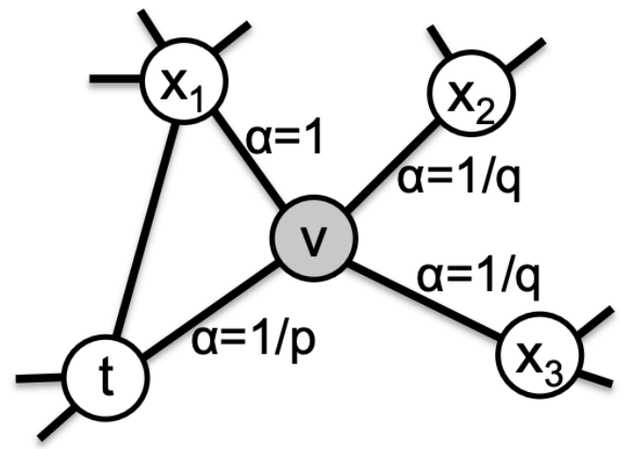


Figure 2: Illustration of the random walk procedure in *node2vec*. The walk just transitioned from t to v and is now evaluating its next step out of node v. Edge labels indicate search biases α.

Flexible notion of neighborhoods

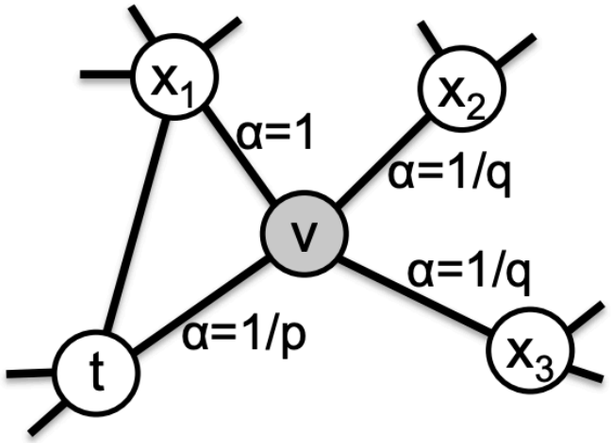


Figure 2: Illustration of the random walk procedure in *node2vec*. The walk just transitioned from t to v and is now evaluating its next step out of node v. Edge labels indicate search biases α.

- Return parameter p :
  - Controls the likelihood of immediately revisiting a node in a walk.
  - high value - less likely to sample already visited
  - low value - lead the walk to backtrack a step (keeping the walk 'local')
- In-out parameter q :
  - Allows the search to differentiate between 'inward' and 'outward' nodes
    - high value - walk is based towards nodes close to node t
    - low value - walk is inclined to visit nodes further away from the node t

Feature Learning Framework

We formulate feature learning in networks as a maximum likelihood optimization problem. Let $G = (V, E)$ be a given network. Our analysis is general and applies to any (un)directed, (un)weighted network. Let $f : V \rightarrow \mathbb{R}^d$ be the mapping function from nodes to feature representaions we aim to learn for a downstream prediction task. Here d is a parameter specifying the number of dimensions of our feature representation. Equivalently, f is a matrix of size $|V| \times d$ parameters. For every source node $u \in V$, we define $N_S(u) \subset V$ as a *network neighborhood* of node u generated through a neighborhood sampling strategy S.

We proceed by extending the Skip-gram architecture to networks [21, 24]. We seek to optimize the following objective function, which maximizes the log-probability of observing a network neighborhood $N_S(u)$ for a node u conditioned on its feature representation, given by f:

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

Feature Learning Framework

In order to make the optimization problem tractable, we make two standard assumptions:

- Conditional independence. We factorize the likelihood by assuming that the likelihood of observing a neighborhood node is independent of observing any other neighborhood node given the feature representation of the source:

$$Pr(N_S(u)|f(u)) = \prod_{n_i \in N_S(u)} Pr(n_i|f(u)).$$

- Symmetry in feature space. A source node and neighborhood node have a symmetric effect over each other in feature space. Accordingly, we model the conditional likelihood of every source-neighborhood node pair as a softmax unit parametrized by a dot product of their features:

$$Pr(n_i|f(u)) = \frac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))}.$$

With the above assumptions, the objective in Eq. 1 simplifies to:

$$\max_f \sum_{u \in V} \left[-\log Z_u + \sum_{n_i \in N_S(u)} f(n_i) \cdot f(u) \right]. \quad (2)$$

The per-node partition function, $Z_u = \sum_{v \in V} \exp(f(u) \cdot f(v))$, is expensive to compute for large networks and we approximate it using negative sampling [22]. We optimize Eq. 2 using stochastic gradient ascent over the model parameters defining the features f.

Feature learning methods based on the Skip-gram architecture have been originally developed in the context of natural language [21]. Given the linear nature of text, the notion of a neighborhood can be naturally defined using a sliding window over consecutive words.

Summary Pseudocode

3.2.3 *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)
$\pi =$ 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**
    $walk =$ node2vecWalk(G', u, l)
    Append $walk$ to $walks$
$f =$ StochasticGradientDescent(k, d, $walks$)
return f

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

Experiments

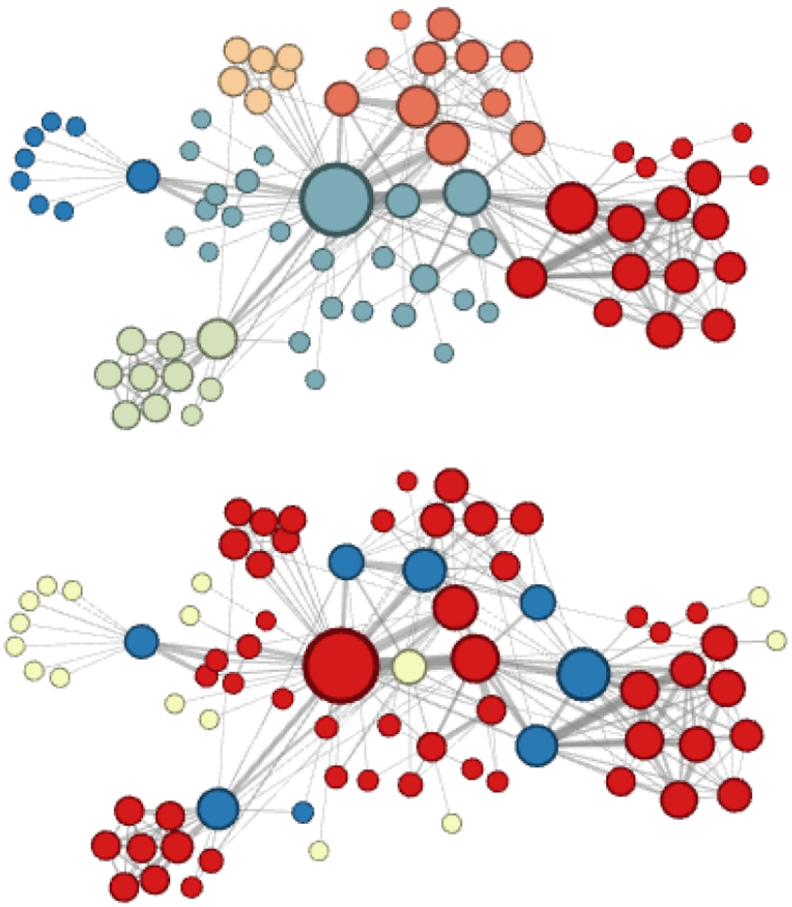


Figure 3: Complementary visualizations of Les Misérables coappearance network generated by *node2vec* with label colors reflecting homophily (top) and structural equivalence (bottom).

- **Case Study : Les Misérables network**
  - Shows both DFS, BFS equivalence notions are commonly exhibited in most real-world networks and have a significant impact on the performance of the learned representations for prediction task

Experiments

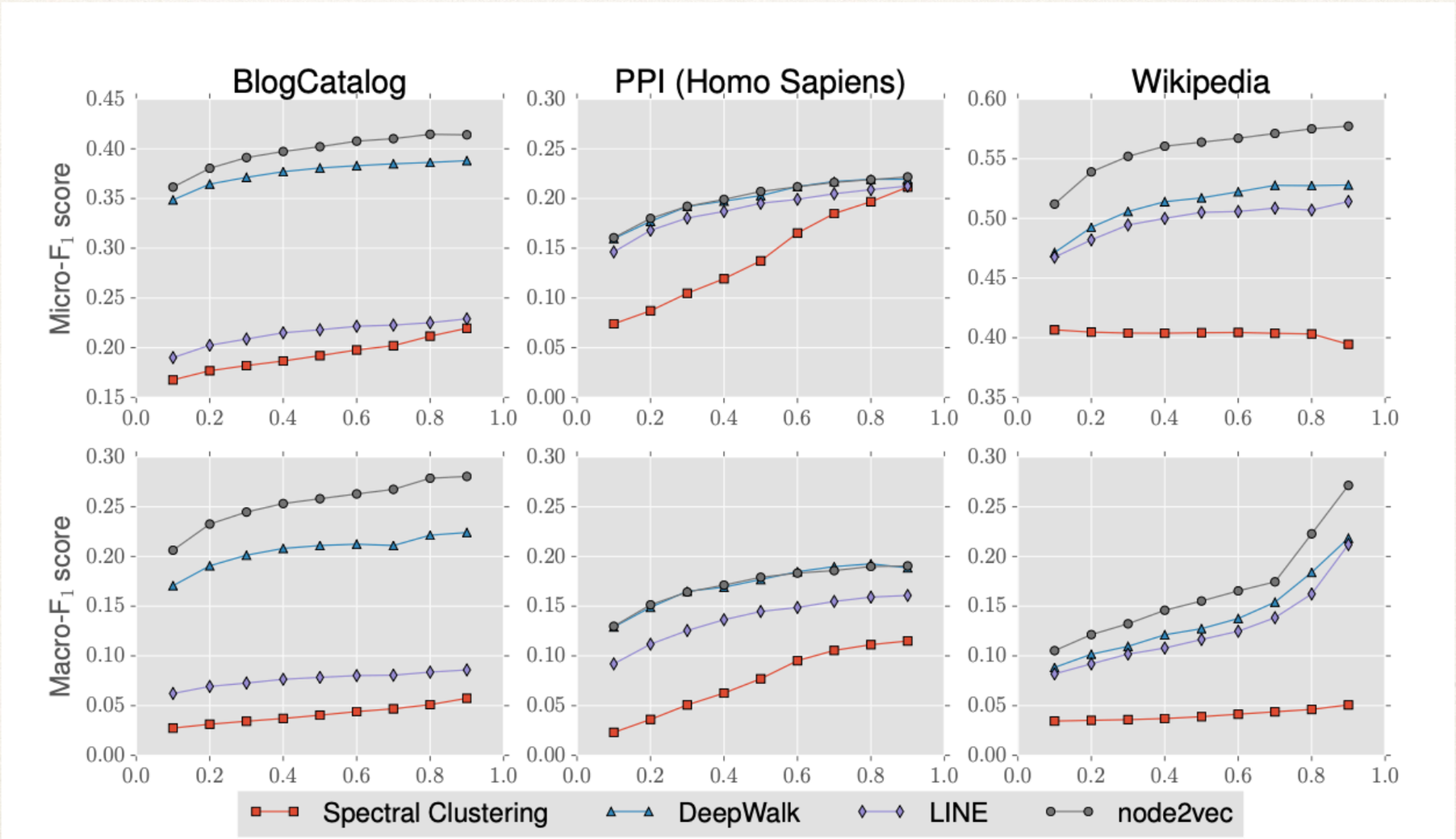
- Node2vec against Spectral clustering, DeepWalk, LINE
- Multi-label classification setting :
  - Every node is assigned one or more labels from a finite set L. During the training phase, we observe a certain fraction of nodes and all their labels. The task is to predict the labels for the remaining nodes.

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**

Table 2: Macro-F_1 scores for multilabel classification on BlogCatalog, PPI (Homo sapiens) and Wikipedia word cooccurrence networks with 50% of the nodes labeled for training.

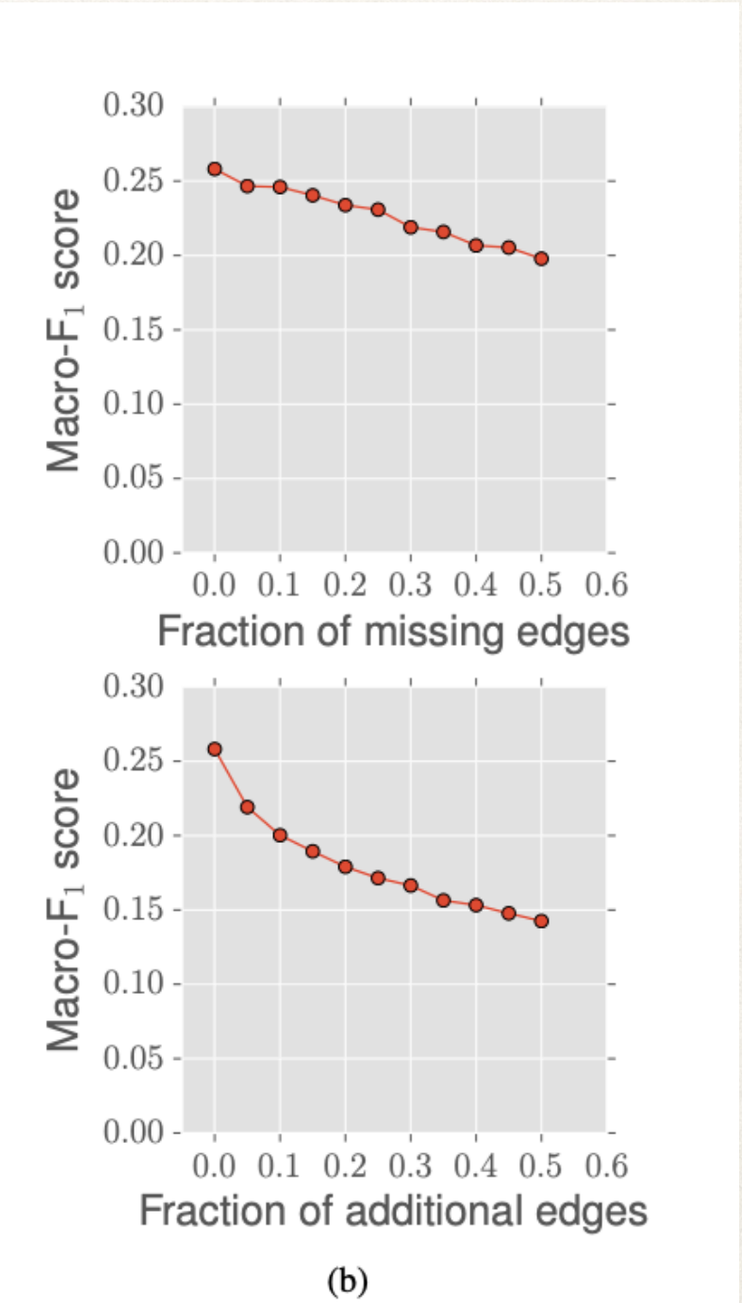
Experiments

- Node2vec against Spectral clustering, DeepWalk, LINE



Perturbation Analysis

- For many real-world networks, we do not have access to accurate information about the network structure. We performed a perturbation study where we analyzed the performance of node2vec for two imperfect information scenarios related to the edge structure in the BlogCatalog network.
- In the first scenario, we measure performance as a function of the fraction of missing edges (relative to the full network).
- In the second perturbation setting, we have noisy edges between randomly selected pairs of nodes in the network.



Scalability

- To test for scalability, we learn node representations using node2vec with default parameter values for Erdos-Renyi graphs with increasing sizes from 100 to 1,000,000 nodes and constant average degree of 10.
- Node2vec scales linearly with increase in number of nodes generating representations for one million nodes in less than four hours.

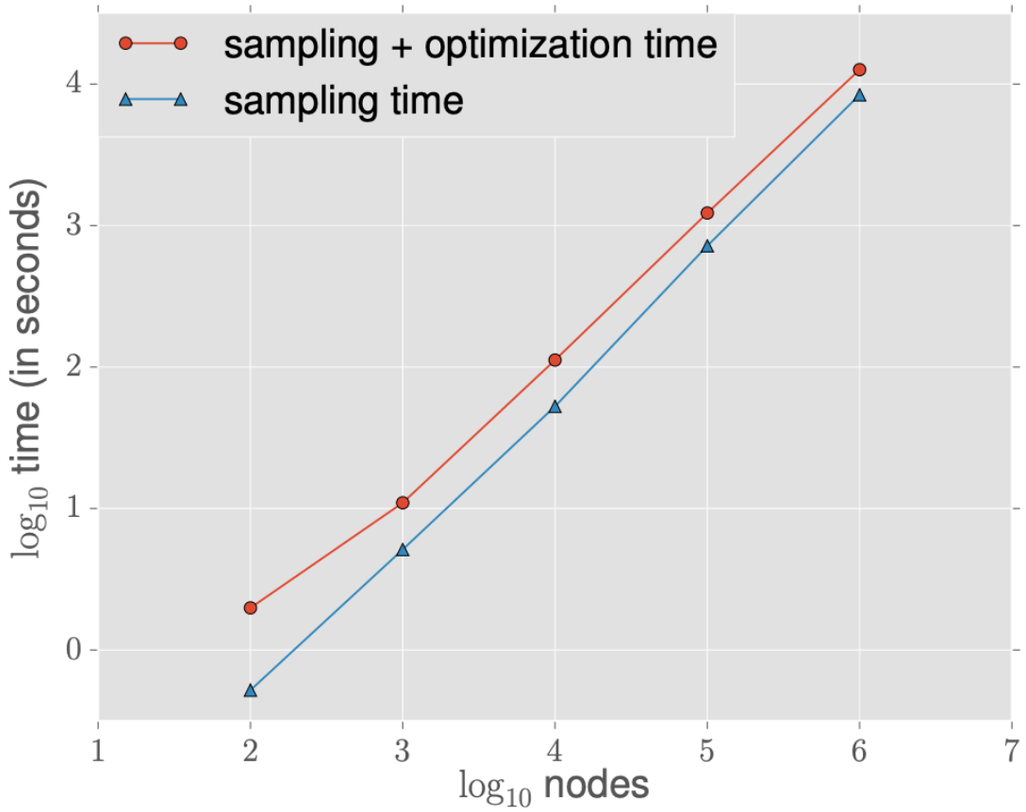


Figure 6: Scalability of *node2vec* on Erdos-Renyi graphs with an average degree of 10.

Link prediction

- In link prediction, we are given a network with a certain fraction of edges removed, and we would like to predict these missing edges.

Op	Algorithm	Dataset		
		Facebook	PPI	arXiv
	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
(a)	Spectral Clustering	0.5960	0.6588	0.5812
	DeepWalk	0.7238	0.6923	0.7066
	LINE	0.7029	0.6330	0.6516
	node2vec	0.7266	0.7543	0.7221
(b)	Spectral Clustering	0.6192	0.4920	0.5740
	DeepWalk	**0.9680**	0.7441	0.9340
	LINE	0.9490	0.7249	0.8902
	node2vec	**0.9680**	**0.7719**	**0.9366**
(c)	Spectral Clustering	0.7200	0.6356	0.7099
	DeepWalk	0.9574	0.6026	0.8282
	LINE	0.9483	0.7024	0.8809
	node2vec	0.9602	0.6292	0.8468
(d)	Spectral Clustering	0.7107	0.6026	0.6765
	DeepWalk	0.9584	0.6118	0.8305
	LINE	0.9460	0.7106	0.8862
	node2vec	0.9606	0.6236	0.8477

Table 4: Area Under Curve (AUC) scores for link prediction. Comparison with popular baselines and embedding based methods bootstapped using binary operators: (a) Average, (b) Hadamard, (c) Weighted-L1, and (d) Weighted-L2 (See Table 1 for definitions).

Conclusion & Discussion

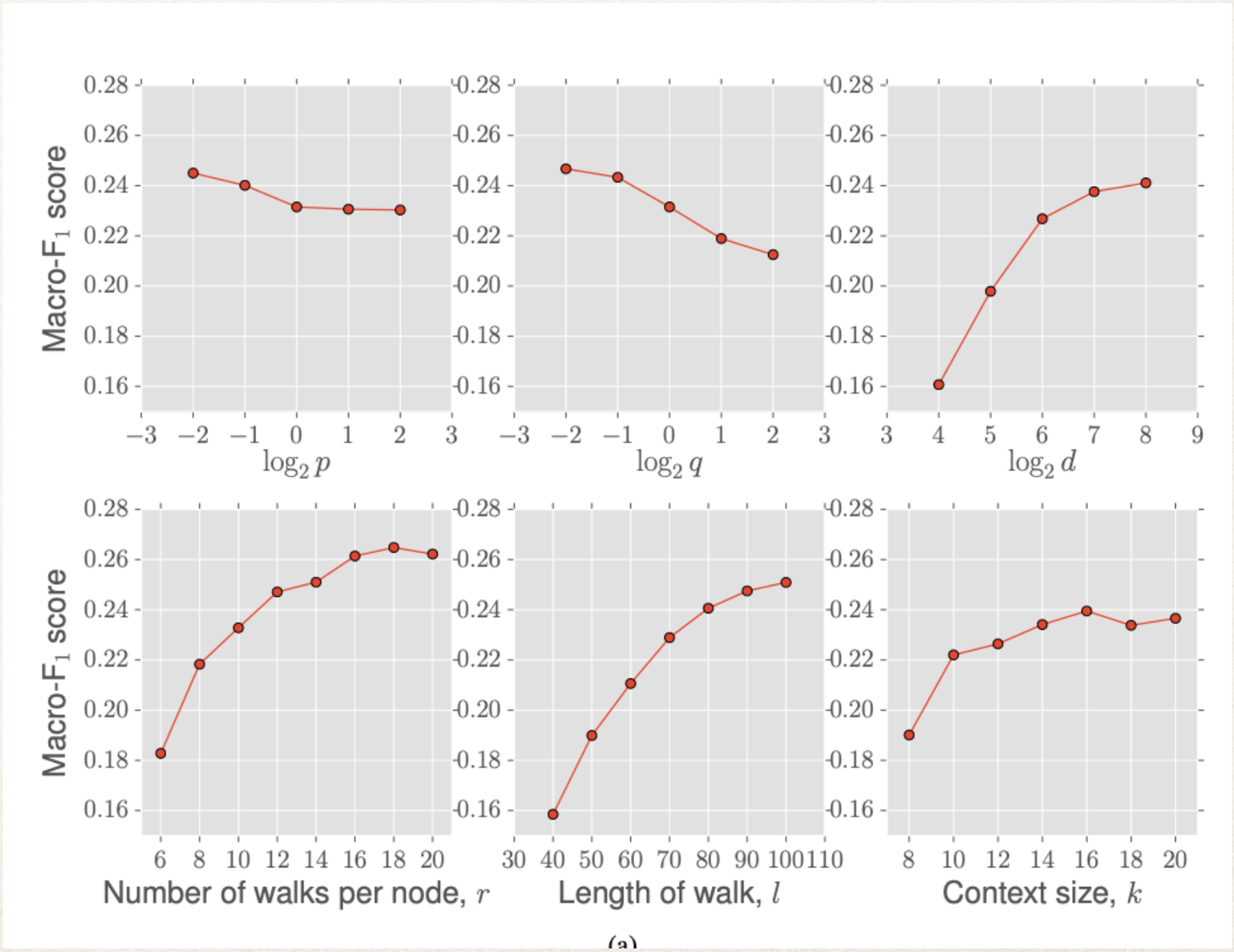
- Node2vec can explain classic search strategies on the basis of the exploration-exploitation trade-off.
  - The search strategy utilized is both flexible and controllable exploring network neighborhoods through parameters p and q.
- Continuous feature representations are the backbone of many deep learning algorithms, and it would be interesting to use node2vec representations as building blocks for end-to-end deep learning on graphs.

References

- Grover, A., Leskovec, J. (2016). Node2vec: Scalable Feature Learning for Networks.
- Grover A. (2016, Oct 11). Node2vec: Scalable Feature Learning for Networks. Youtube. https://www.youtube.com/watch?v=1_QH5BEP5BM&ab_channel=KDD2016video
- Cohen, E. (2018, Nov 23). PyData Tel Aviv Meetup: Node2vec - Elior Cohen. Youtube. https://www.youtube.com/watch?v=828rZgV9t1g&ab_channel=PyData

Appendix

- Parameter Sensitivity



(a)