node2vec: Scalable Feature Learning for Networks

Reported by: Xinyi Wang

1.Why we come up with this work:
present feature learning approaches are not expressive enough to capture the diversity of connectivity patterns observed in networks.

2.Objective function:
In node2vec,we learn a mapping of nodes to a low-dimensional space of features that maximizes the likelihood of preserving network neighborhoods of nodes.

3.Definition of neighborhood:
We define a flexible notion of a node’s network neighborhood and design a biased random walk procedure, which efficiently explores diverse neighborhoods.

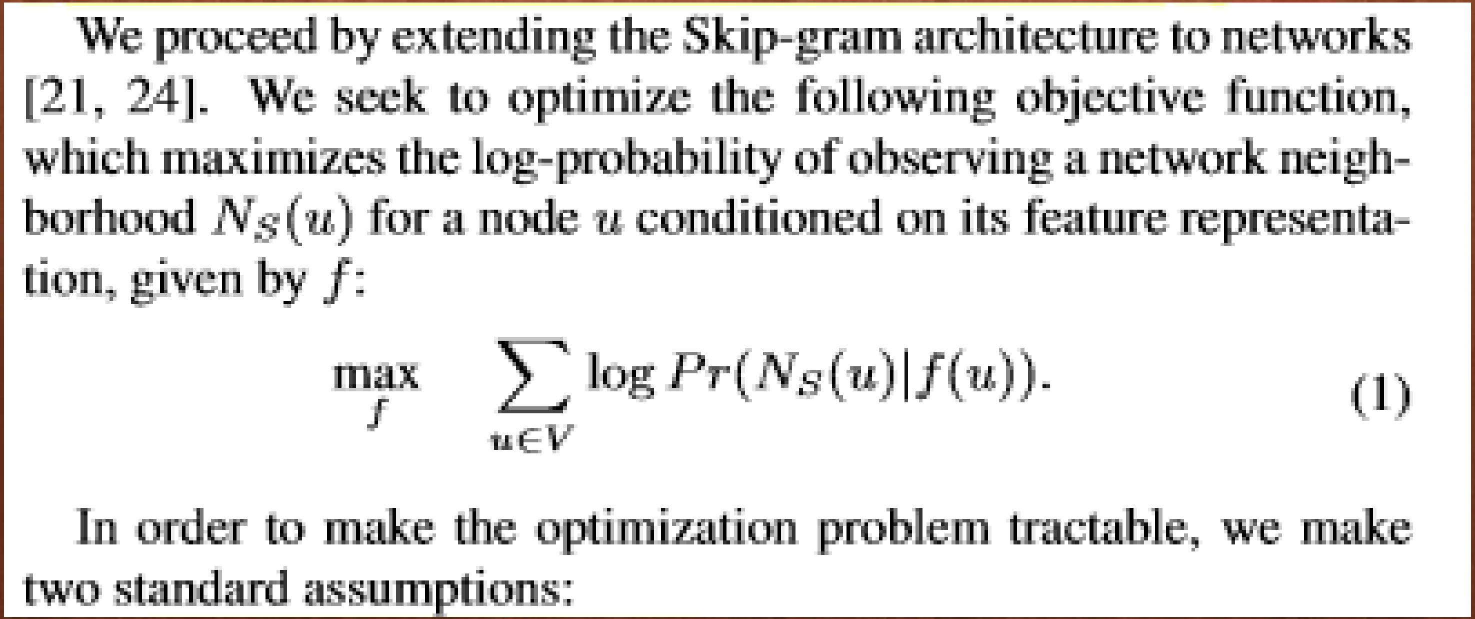
4.Experiments:
We demonstrate the efficacy of node2vec over existing state-of the-art techniques on multi-label classification and link prediction in several real-world networks from diverse domains.

1.Why we come up with this work:

Inspired by the Skip-gram model, recent research established an analogy for networks by representing a network as a “document” [24, 28]. The same way as a document is an ordered sequence of words, one could sample sequences of nodes from the underlying network and turn a network into a ordered sequence of nodes. However, there are many possible sampling strategies for nodes, resulting in different learned feature representations. In fact, as we shall show, there is no clear winning sampling strategy that works across all networks and all prediction tasks. This is a major shortcoming of prior work which fail to offer any flexibility in sampling of nodes from a network [24, 28]. Our algorithm node2vec overcomes this limitation by designing a flexible objective that is not tied to a particular sampling strategy and provides parameters to tune the explored search space.

2.Objective function:

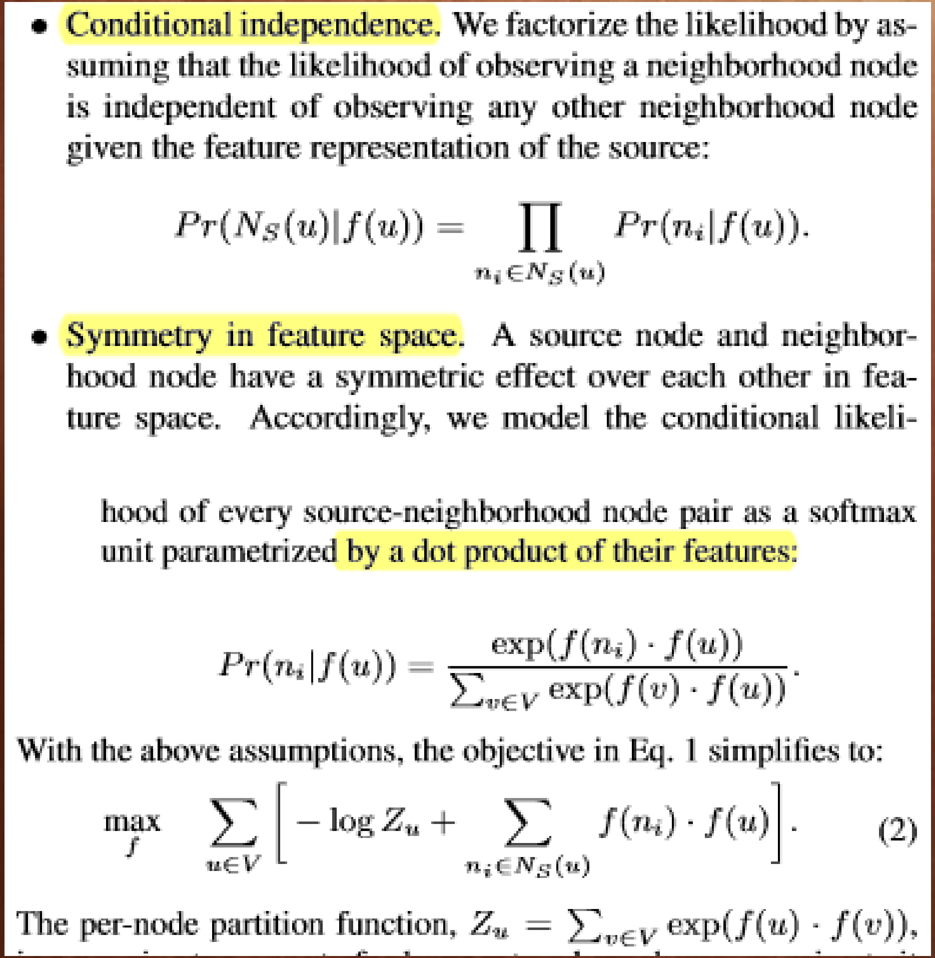
Intuitively, our approach returns feature representations that maximize the likelihood of preserving network neighborhoods of nodes in a d-dimensional feature space. We define NS(u) ⊂ V as a network neighborhood of node u generated through a neighborhood sampling strategy S.



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:

$$\max_f \sum_{u \in V} \log Pr(N_S(u)|f(u)). \tag{1}$$

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:

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

The per-node partition function, $Z_u = \sum_{v \in V} \exp(f(u) \cdot f(v))$,

In order to make the optimization problem tractable, we make two standard assumptions.
Furthermore, negative sampling and stochastic gradient is used to optimize equation(2)

3.Definition of neighborhood:

We view the problem of sampling neighborhoods of a source node as a form of local search. Figure 1 shows a graph, where given a source node u we aim to generate (sample) its neighborhood NS(u). Importantly, to be able to fairly compare different sampling strategies S, we shall constrain the size of the neighborhood set NS to k nodes and then sample multiple sets for a single node u. Generally, there are two extreme sampling strategies for generating neighborhood set(s) NS of k nodes:

- Breadth-first Sampling(BFS) The neighborhood NS is restricted to nodes which are immediate neighbors of the source. For example, in Figure 1 for a neighborhood of size k = 3, BFS samples nodes s1, s2, s3.
- Depth-first Sampling (DFS) The neighborhood consists of nodes sequentially sampled at increasing distances from the source node. In Figure 1, DFS samples s4, s5, s6.

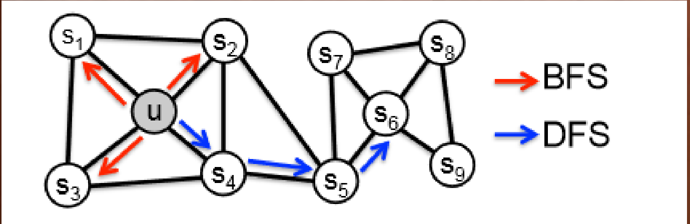


Figure 1: BFS and DFS search strategies from node u ($k = 3$).

Then, let’s introduce two hypothesis:

Under the homophily hypothesis [7, 36] nodes that are highly interconnected and belong to similar network clusters or communities should be embedded closely together. (e.g., nodes s1 and u in Figure 1 belong to the same network community)

under the structural equivalence hypothesis [10] nodes that have similar structural roles in networks should be embedded closely together (e.g., nodes u and s6 in Figure 1 act as hubs of their corresponding communities). Importantly, unlike homophily, structural equivalence does not emphasize connectivity; nodes could be far apart in the network and still have the same structural role.

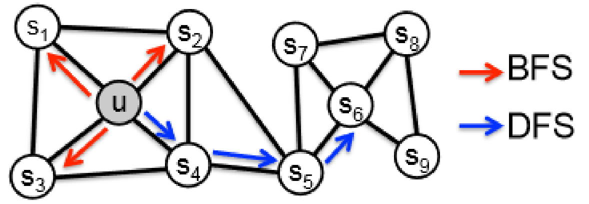
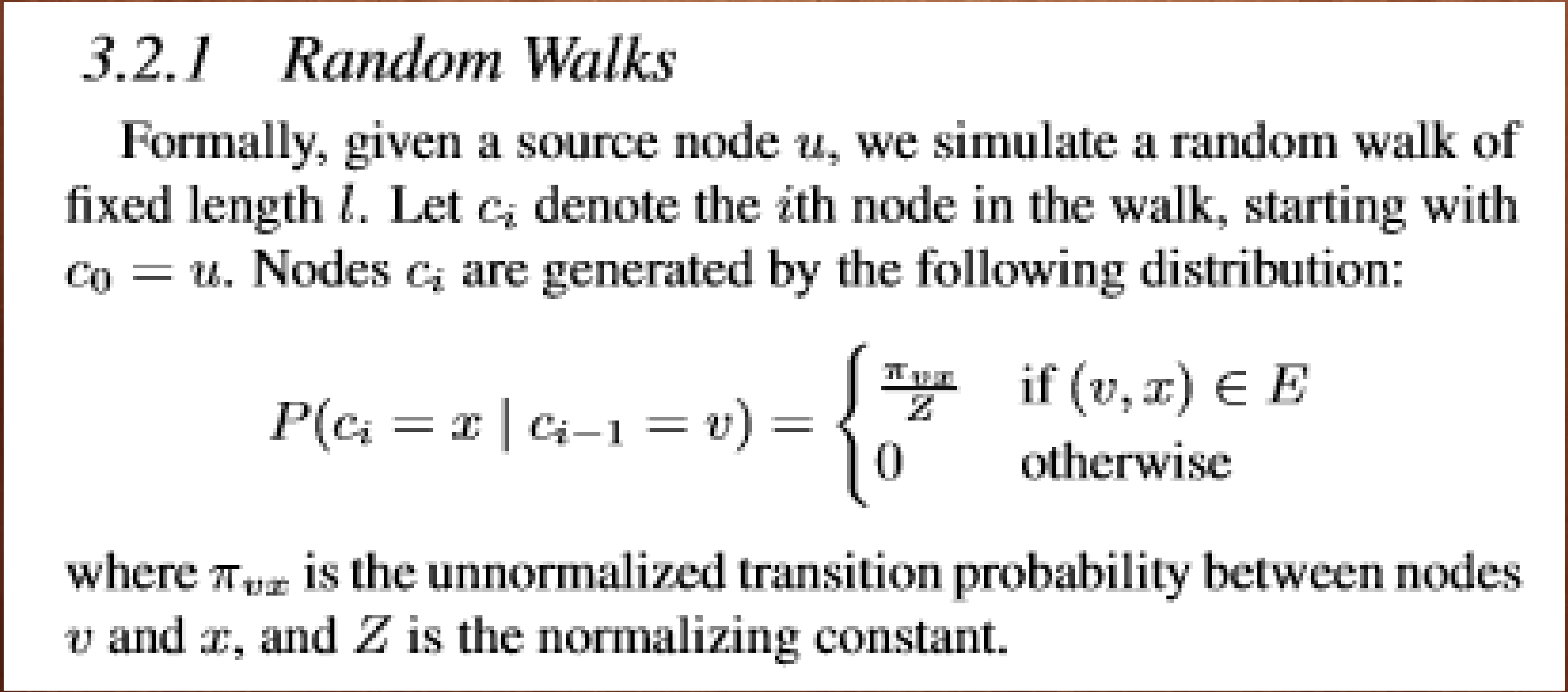


Figure 1: BFS and DFS search strategies from node u ($k = 3$).

We observe that BFS and DFS strategies play a key role in producing representations that reflect either of the above equivalences. In particular, the neighborhoods sampled by BFS lead to embeddings that correspond closely to structural equivalence By restricting search to nearby nodes, BFS achieves this characterization and obtains a microscopic view of the neighborhood of every node. Additionally, in BFS, nodes in the sampled neighborhoods tend to repeat many times. This is also important as it reduces the variance in characterizing the distribution of 1-hop nodes with respect the source node(?). However, a very small portion of the graph is explored for any given k.

The opposite is true for DFS which can explore larger parts of the network as it can move further away from the source node u (with sample size k being fixed). In DFS, the sampled nodes more accurately reflect a macro-view of the neighborhood which is essential in inferring communities based on homophily.

3.2 node2vec Building on the above observations, we design a flexible neighborhood sampling strategy which allows us to smoothly interpolate between BFS and DFS. We achieve this by developing a flexible biased random walk procedure that can explore neighborhoods in a BFS as well as DFS fashion.



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

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

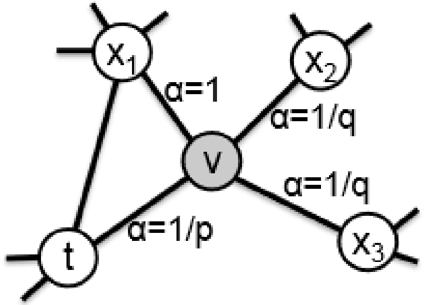
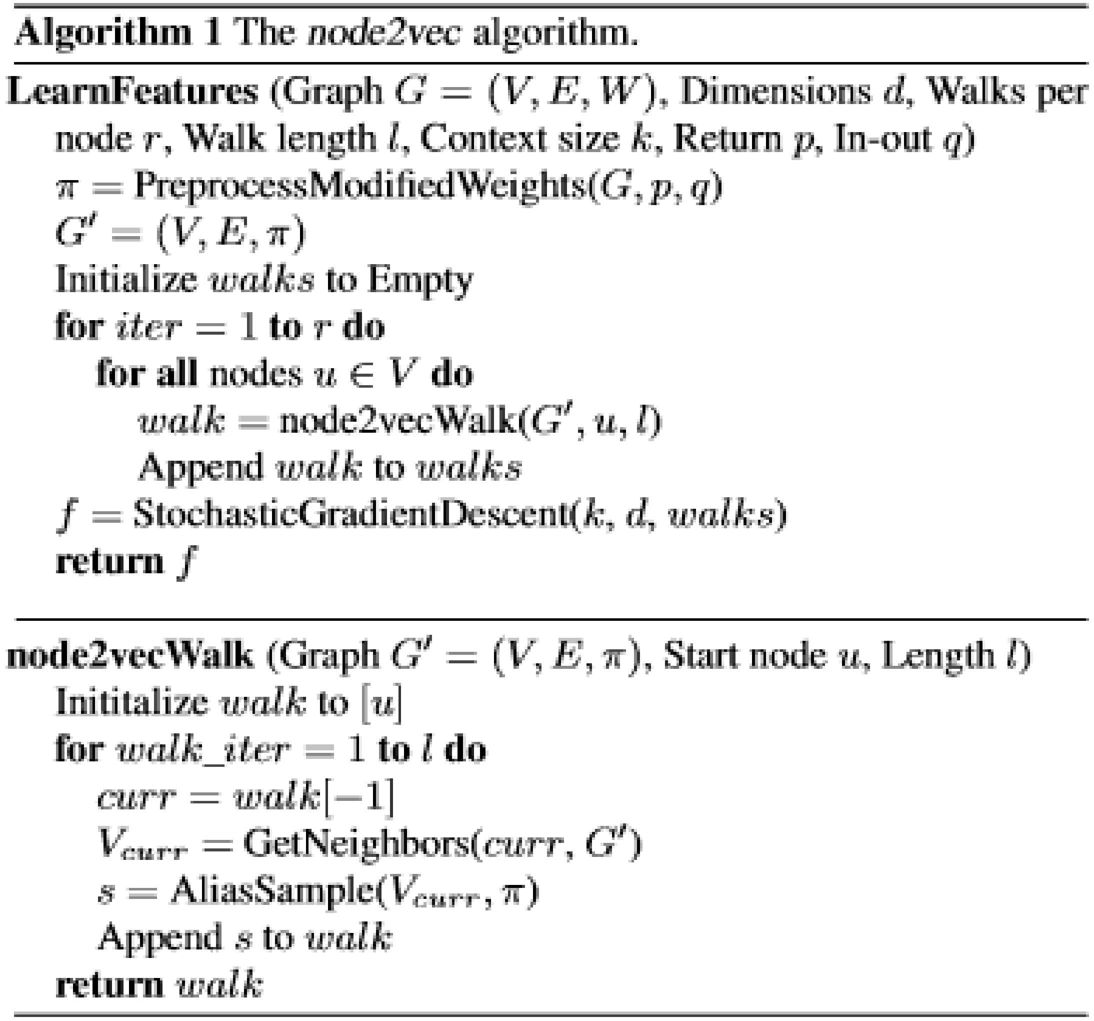


Figure 2: Illustration of the random walk procedure in *node2vec*. The walk just transitioned from t to v and is now evaluating its next step out of node v. Edge labels indicate search biases α.

Return parameter, p. Parameter p controls the likelihood of immediately revisiting a node in the walk. Setting it to a high value (> max(q,1))ensures that we are less likely to sample an already visited node in the following two steps (unless the next node in the walk had no other neighbor). This strategy encourages moderate exploration and avoids 2-hop redundancy in sampling. On the other hand, if p is low (< min(q,1)), it would lead the walk to backtrack a step (Figure 2) and this would keep the walk “local” close to the starting node u.

In-out parameter, q. Parameter q allows the search to differentiate between “inward” and “outward” nodes. Going back to Figure 2, if $q > 1$, the random walk is biased towards nodes close to node t. Such walks obtain a local view of the underlying graph with respect to the start node in the walk and approximate BFS behavior in the sense that our samples comprise of nodes within a small locality. In contrast, if $q < 1$, the walk is more inclined to visit nodes which are further away from the node t. Such behavior is reflective of DFS which encourages outward exploration. However, an essential difference here is that we achieve DFS-like exploration within the random walk framework. Hence, the sampled nodes are not at strictly increasing distances from a given source node u, but in turn, we benefit from tractable preprocessing and superior sampling efficiency of random walks. Note that by setting πv,x to be a function of the preceding node in the walk t, the random walks are 2nd order Markovian.

Pseudo code



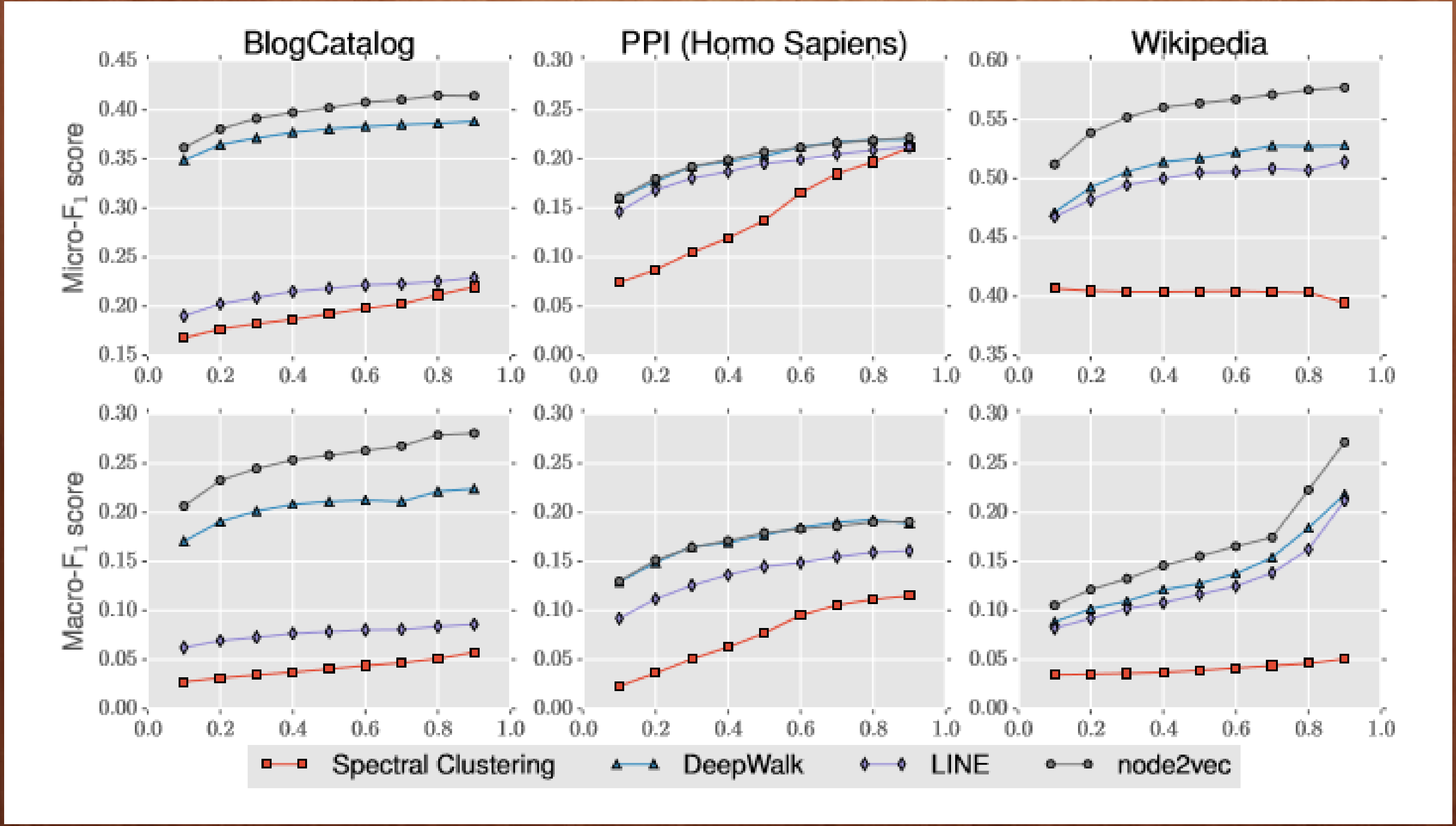
Algorithm 1 The *node2vec* algorithm.

```
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($V_{curr}, \pi$)
        Append $s$ to $walk$
    return $walk$
```

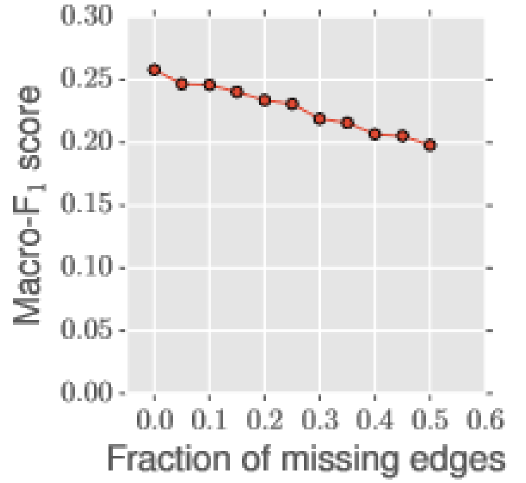
4.Experiment results

4.1comparative experiments

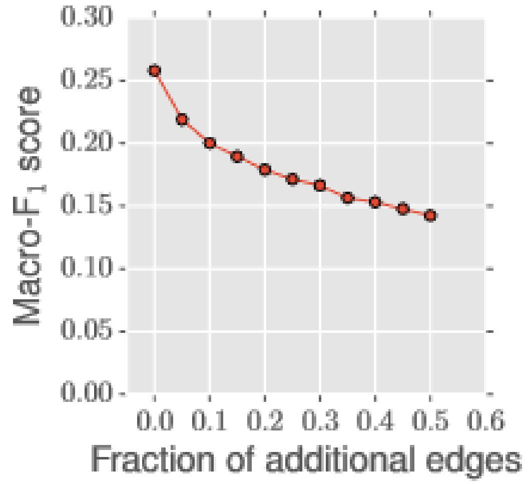


From the results, it is evident we can see how the added flexibility in exploring neighborhoods allows node2vec to outperform the other benchmark algorithms. In Blog Catalog, we can discover the right mix of homophily and structural equivalence by setting parameters p and q to low values, giving us 22.3% gain over Deep Walk and 229.2% gain over LINE in Macro-F1 scores. LINE showed worse performance than expected, which can be explained by its inability to reuse samples(?), a feat that can be easily done using the random walk methods.

Perturbation Analysis



we measure performance as a function of the fraction of missing edges (relative to the full network). The missing edges are chosen randomly, subject to the constraint that the number of connected components in the network remains fixed. As we can see in Figure 5b(top), the decreaseinMacro-F1 score as the fraction of missing edges increases is roughly linear with a small slope.



we have noisy edges between randomly selected pairs of nodes in the network. As shown in Figure 5b(bottom), the performance of node2vec declines slightly faster initially when compared with the setting of missing edges, however, the rate of decrease in Macro-F1 score gradually slows down over time

Thank you