# node2vec: Scalable Feature Learning for Networks

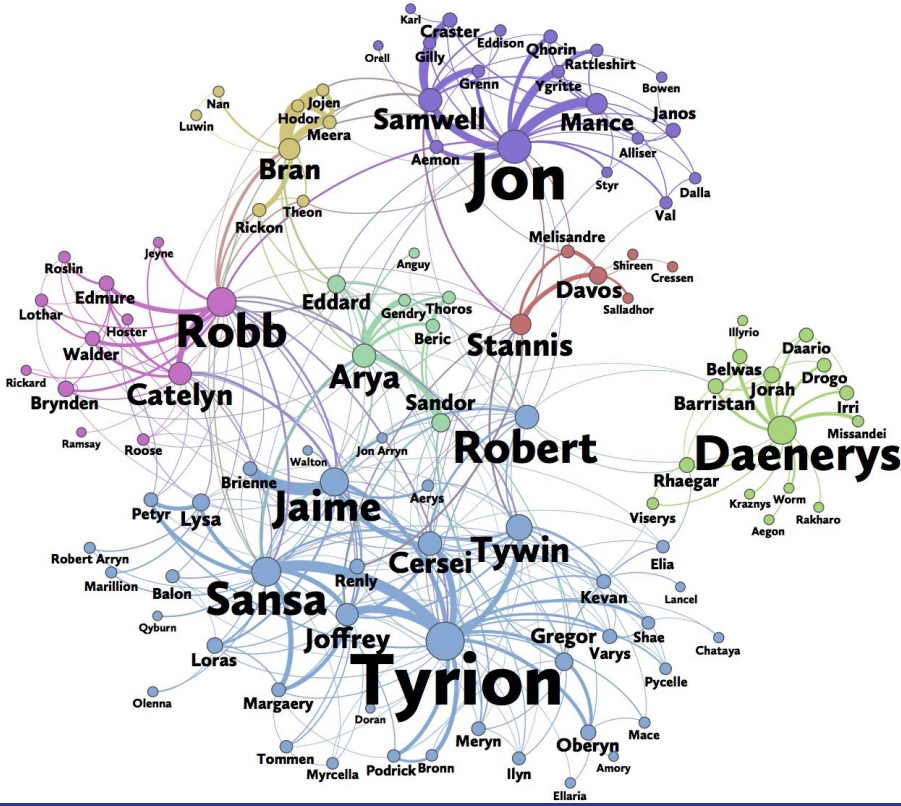
Presenter: Tomáš Nováček, Faculty of information technology, CTU
Supervisor: doc. RNDr. Ing. Marcel Jiřina, Ph.D.

Authors: Aditya Grover, Jure Leskovec; Stanford University

# Background

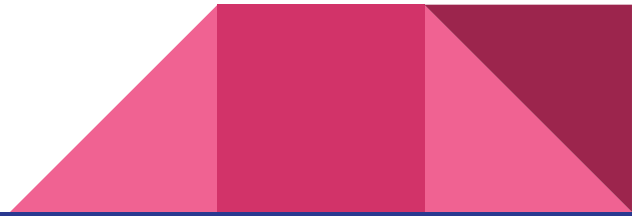
# Tasks in network analysis

- Labels prediction
  - e. g. is user interested in Game of Thrones?
- Link prediction
  - e. g. are users real-life friends?
- Community detection
  - e. g. do characters in a book often meet?



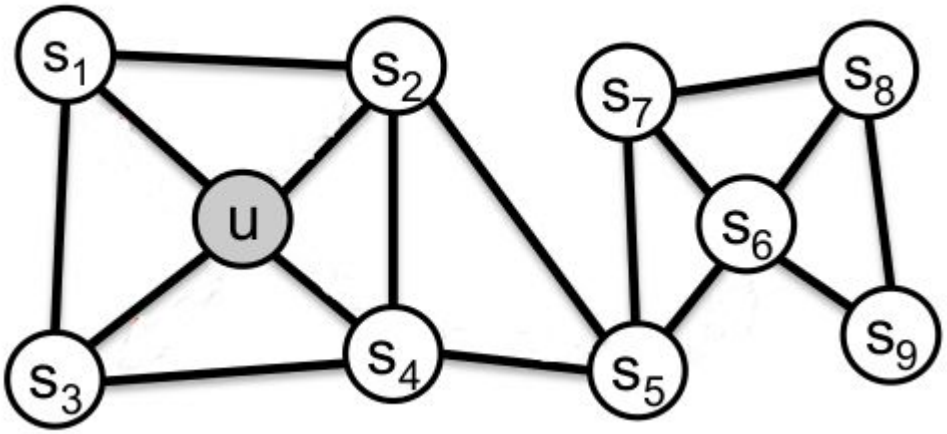
# Feature learning

1. Hand-engineering features
    - Based on expert knowledge
    - - Time-consuming
    - - Not generic enough
2. Solving optimization problem
    - Supervised
        - Good accuracy, high training time
    - Unsupervised
        - Efficient, hard to find the objective
    - Trade-off in efficiency and accuracy



# Optimization problem

- Classic approach – linear and non-linear dimensionality reduction
- Alternative approach – preserving local neighbours
  - Most attempts rely on rigid notion
  - Insensitivity to connectivity patterns
    - Homophily
      - Based on communities
    - Structural equivalence
      - Roles in network
      - Equivalence does not emphasise connectivity



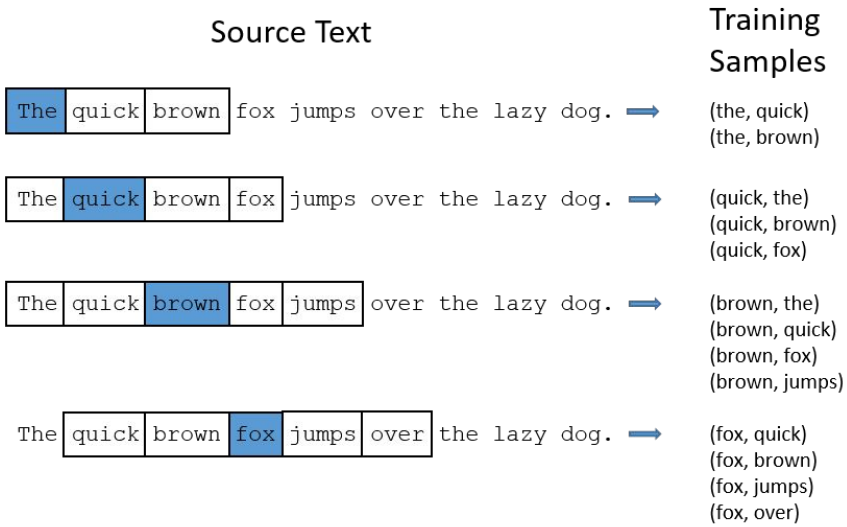
# node2vec

# node2vec

- Semi-supervised algorithm
- Generates sample network neighbours
  - Maximises likelihood of preserving neighborhood
  - Flexible notion of neighborhood of nodes
- Tunable parameters
  - Unsupervised
  - Semi-supervised
- Parallelizable

# Skip-gram model

- Made for NLP (word2vec)
- Prediction of consecutive words
  - Similar context => similar meaning
- Learning feature representations
  - Optimizing likelihood objective
  - Neighborhood preserving
- Can we use it for networks?
  - Yes! We have to linearize the network.



# Feature learning in networks

- $G = (V, E)$
  - V – vertices (nodes)
  - E – edges (links)
  - (un)directed (un)weighted
- $f : V \rightarrow R^d$
  - f – mapping func from nodes to feature representations
  - d – number of dimensions
  - matrix of size $|V| \times d$ parameters
- $\forall u \in V: N_S(u) \subset V$
  - $N_S(u)$ – network neighborhood of *u*
  - S – sampling strategy

# Optimizing objective function

- Maximizes the log-probability of observing a network neighborhood

$$\max_{f} \sum_{u \in V} \log Pr(N_S(u)|f(u)) \qquad (1)$$

  - $N_S(u)$ is network neighborhood of node *u*
  - Conditioned on its feature representation, given by *f*

# Assumptions

- Conditional independence
  - Likelihood of observing a neighborhood node is independent of observing any other neighborhood node

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

- Symmetry in feature space
  - Source node and neighborhood node have a symmetric effect over each other

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

# Optimizing objective function

- Thus we can simplify (1):

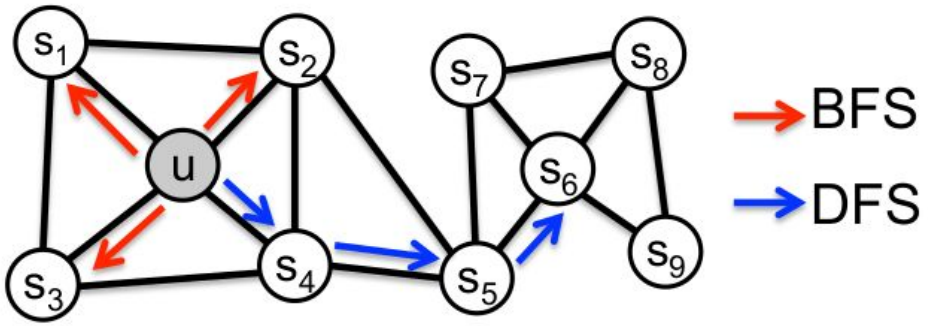
$$\max_f \sum_{u \in V} \left[ -\log Z_u + \sum_{n_i \in N_S(u)} f(n_i) \cdot f(u) \right] \qquad (2)$$

  - Where $Z_u$ is the per-node partition function

    $Z_u = \sum_{v \in V} \exp(f(u) \cdot f(v))$

  - $Z_u$ is approximated using negative sampling

# Search strategies



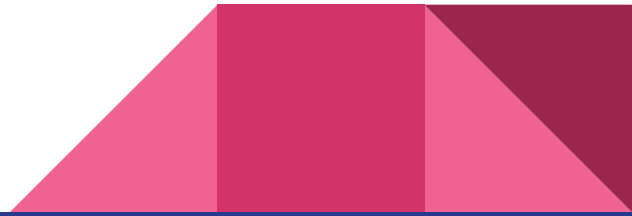

- **Breadth-first sampling (BFS)**
  - Immediate neighbors
  - Small portion of the graph
  - Used by LINE algorithm
- **Depth-first sampling (DFS)**
  - Sequential nodes at increasing distances
  - Larger portion of the graph
  - Used by DeepWalk algorithm
- Constrained size *k*
- Multiple sets for a node

# Breadth-first sampling

- Samples correspond closely to structural equivalence
- Accurate characterization of the local neighborhoods
  - Bridges
  - Hubs
- Nodes tend to repeat
- Small graph is explored
  - Microscopic view of the neighborhood

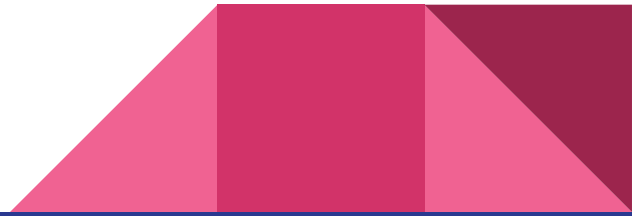
# Depth-first sampling

- Larger part is explored
  - Reflects the macroscopic view
- Can be user to infer homophily
- Need to infer dependencies and their nature
  - High variance
  - Complex dependencies



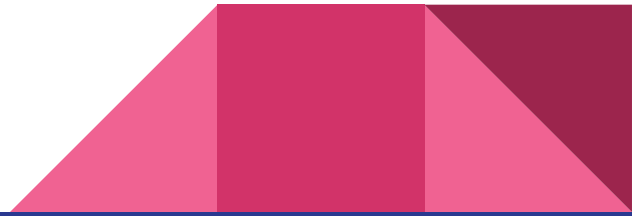
# node2vec

- Flexible biased 2$^{nd}$ order random walk
  - Can return to previously visited node
  - Time and space efficient
- Combines BFS and DFS
  - Controlled by parameters



# Parameters

- Parameter *p* (return parameter)
  - Likelihood of immediately revisiting a node
  - High value (> max(q, 1)) => less probability
  - Low value (< min(q, 1)) => local walk
- Parameter *q* (in-out parameter)
  - Inward vs. outward nodes
  - q > 1
    - Biased to local view of graph
    - BFS-like behaviour
  - q < 1
    - Further nodes
    - DFS-like behaviour

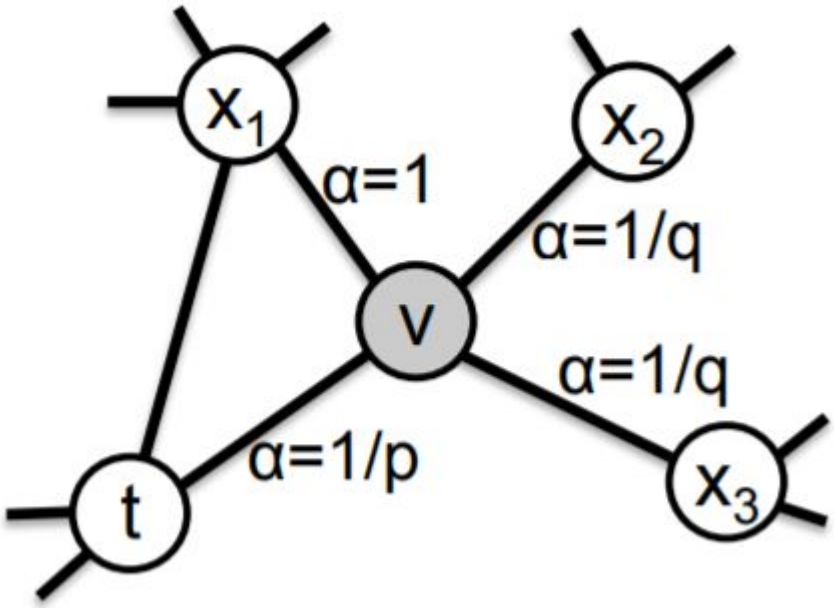


# Search bias

- Edge weights bias
  - Does not account structure
  - Does not combine BFS and DFS
- Parameters p and q

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

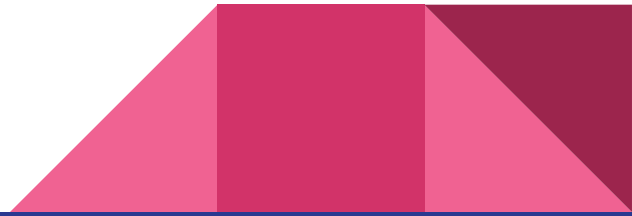
- $\pi_{vx} = \alpha_{pq}(t, x) * w_{vx}$



# node2vec phases

1. Preprocessing to compute transition probabilities
2. Random walk simulations
    - *r* random walks of fixed length *l* from every node
        - Offset of start node implicit bias
3. Optimization using SGD

- Phases executed sequentially
- Phases asynchronous and parallelizable



# Learning edge features

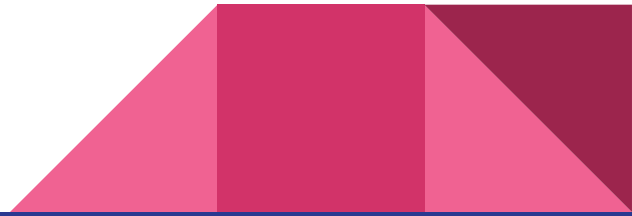
- Binary operator ◦ over corresponding feature vectors f(u) and f(v)
- g(u, v) such that g : V × V → $R^d$

| Operator | Symbol | Definition |
|---|---|---|
| Average | $\boxplus$ | $[f(u) \boxplus f(v)]_i = \frac{f_i(u)+f_i(v)}{2}$ |
| Hadamard | $\boxdot$ | $[f(u) \boxdot f(v)]_i = f_i(u) * f_i(v)$ |
| Weighted-L1 | $\|\cdot\|_{\bar{1}}$ | $\|f(u) \cdot f(v)\|_{\bar{1}i} = \|f_i(u) - f_i(v)\|$ |
| Weighted-L2 | $\|\cdot\|_{\bar{2}}$ | $\|f(u) \cdot f(v)\|_{\bar{2}i} = \|f_i(u) - f_i(v)\|^2$ |

# Experiments

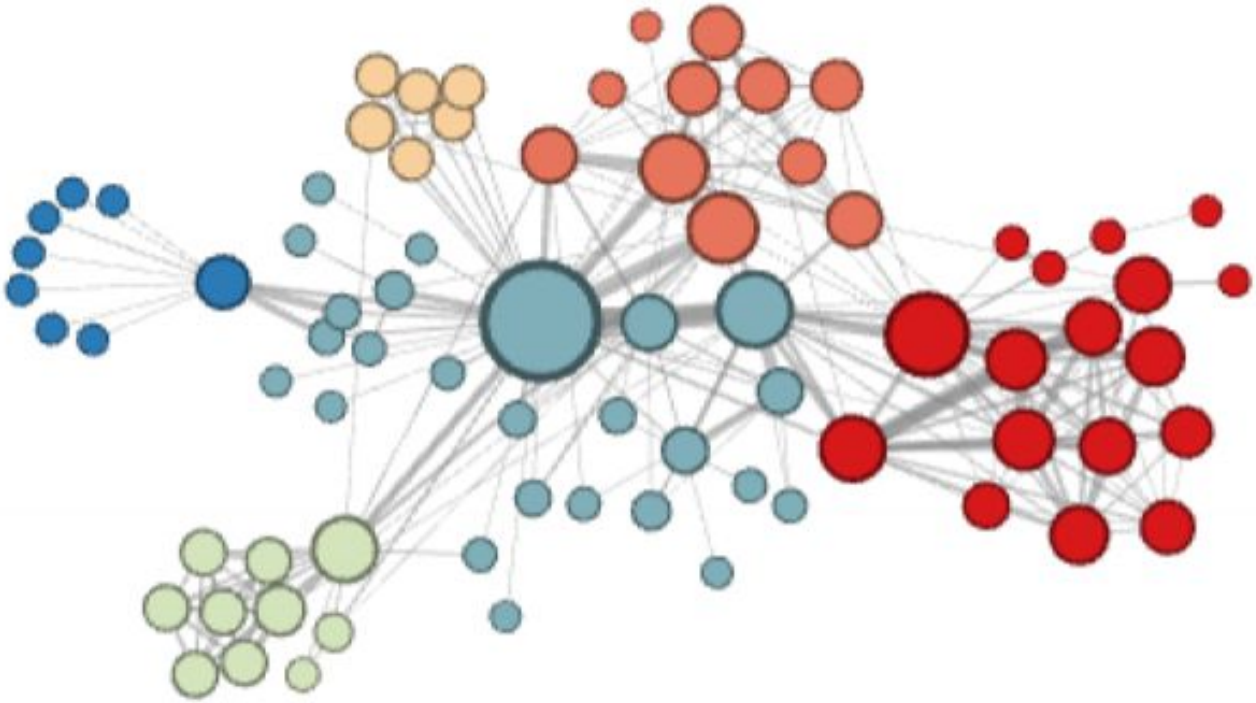
# Les Misérables

- Victor Hugo novel (1862)
- 77 nodes
  - characters from the novel
- 254 edges
  - co-appearing characters
- $d = 16$
  - number of dimensions



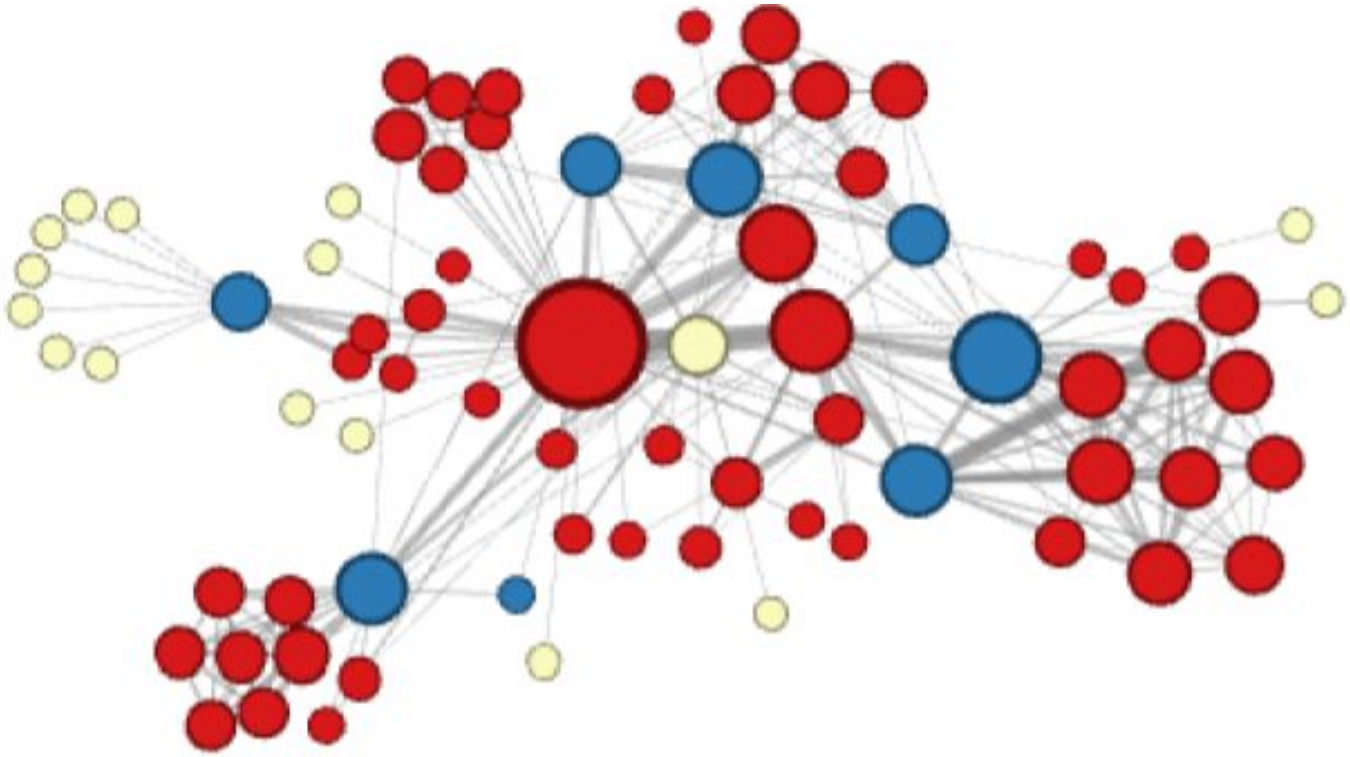
# Les Misérables – homophily

- p = 1
  - less likely to immediately return
- q = 0.5
  - DFS



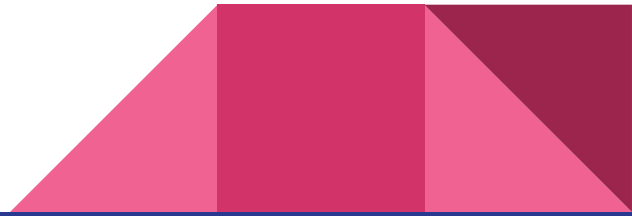
# Les Misérables – structural equivalence

- p = 1
  - likely return
- q = 2
  - BFS



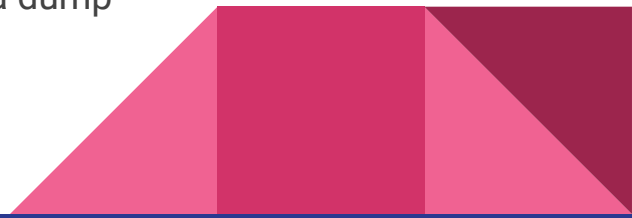
# Benchmark

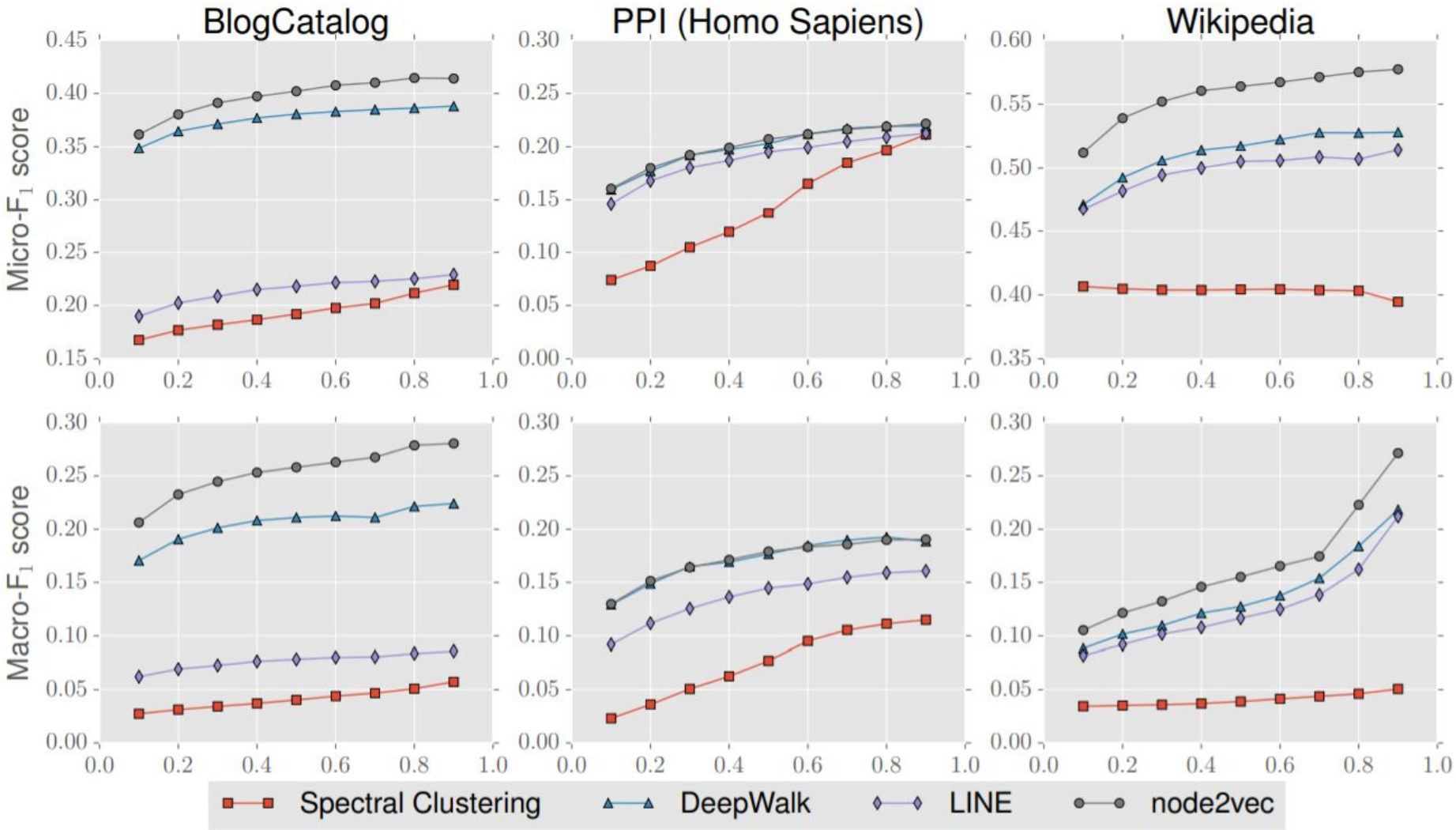
- **Spectral clustering**
  - matrix factorization approach
- **DeepWalk**
  - simulating uniform random walks
  - special case of node2vec with p = 1 and q = 1
- **LINE**
  - first phase – d/2 dimensions, BFS-style simulations
  - second phase – d/2 dimensions, nodes at 2-hop distance from the source
- **node2vec**
  - d = 128, r = 10, l = 80, k = 10
  - p, q learned on 10% labeled data from {0.25, 0.50, 1, 2, 4}

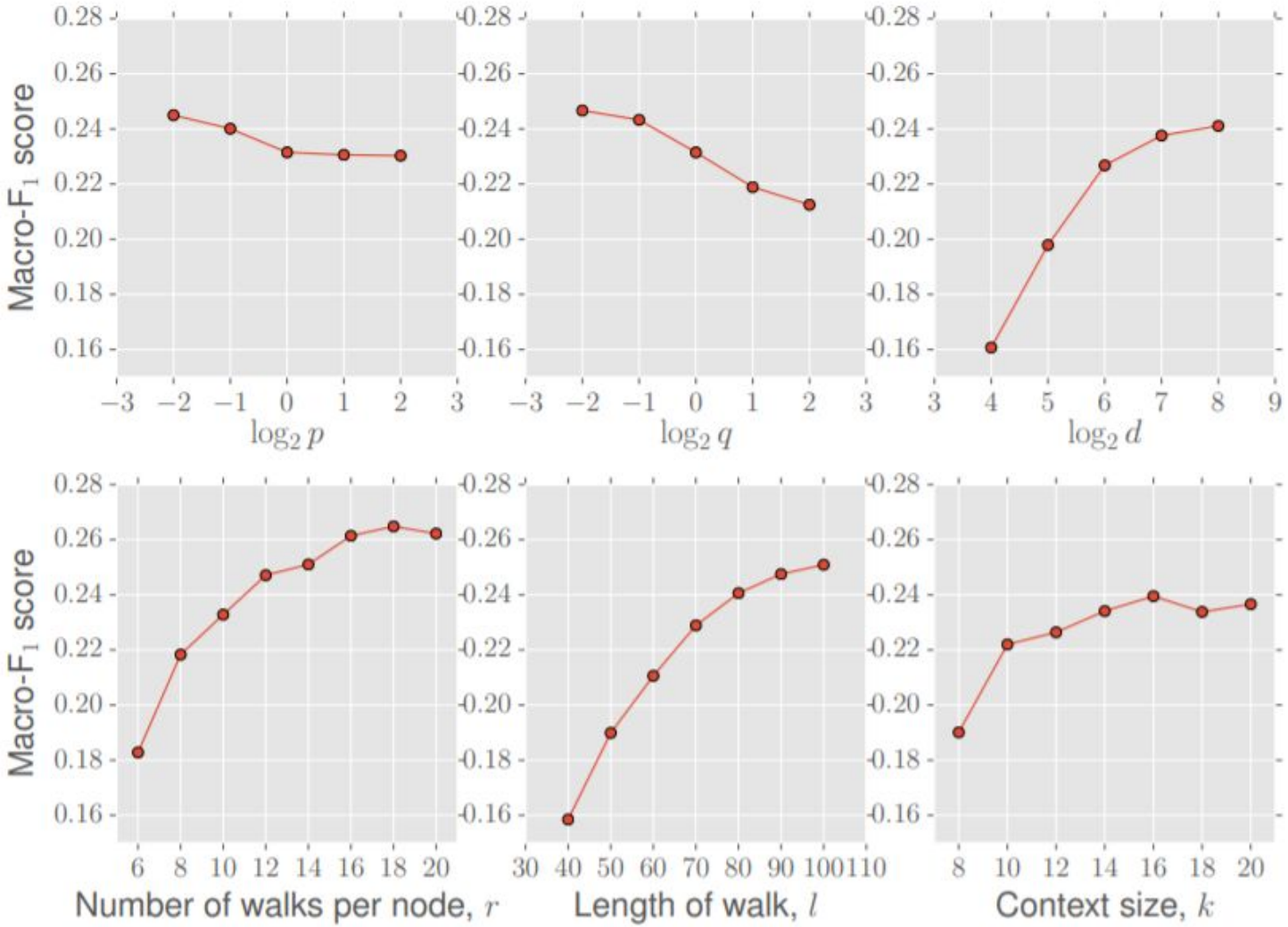


# Datasets

- **BlogCatalog**
  - social relationships of bloggers
  - labels are interests of bloggers
  - 10 312 nodes, 333 983 edges, 39 different labels
- **Protein-Protein Interactions (PPI)**
  - PPI network for Homo sapiens
  - labels from the hallmark gene set
  - 3 890 nodes, 76 584 edges, 50 different labels
- **Wikipedia**
  - co-occurrence of words the first million bytes of the Wikipedia dump
  - labels represent the Part-of-Speech (POS) tags
  - 4 777 nodes, 184 812 edges, 40 different labels




BlogCatalog
PPI (Homo Sapiens)
Wikipedia
Micro-F1 score
Macro-F1 score
Spectral Clustering
DeepWalk
LINE
node2vec


Macro-F1 score
0.28
0.26
0.24
0.22
0.20
0.18
0.16
−3 −2 −1 0 1 2 3
log2 p
log2 q
3 4 5 6 7 8 9
log2 d
6 8 10 12 14 16 18 20
Number of walks per node, r
30 40 50 60 70 80 90 100 110
Length of walk, l
8 10 12 14 16 18 20
Context size, k

# Link prediction

- Generated dataset
  - Positive sample generation
    - randomly removing 50% of edges
    - network stays connected
  - Negative sample generation
    - 50% node pairs
    - no edge between them
- Benchmarks
  - Facebook users (4 039 nodes, 88 234 edges)
  - Protein-Protein Interactions (19 706 nodes and 390 633 edges)
  - arXiv ASTRO-PH (18 722 nodes and 198 110)

| 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 |

# Conclusion

- Efficient scalable algorithm for feature learning
  - both nodes and edges between them
- Network-aware
  - homophily and structural equivalence
- Parameterizable
  - dimensions, length of walk, number of walks, sample size
  - return parameter
  - inward-outward parameter
- Parallelizable
- Link prediction

# Drawbacks

- Vague definitions
- Only works for single-layered networks
- Worse results in dense graphs
- Unanswered questions
  - What if the graph changes?
  - How about featureless nodes?