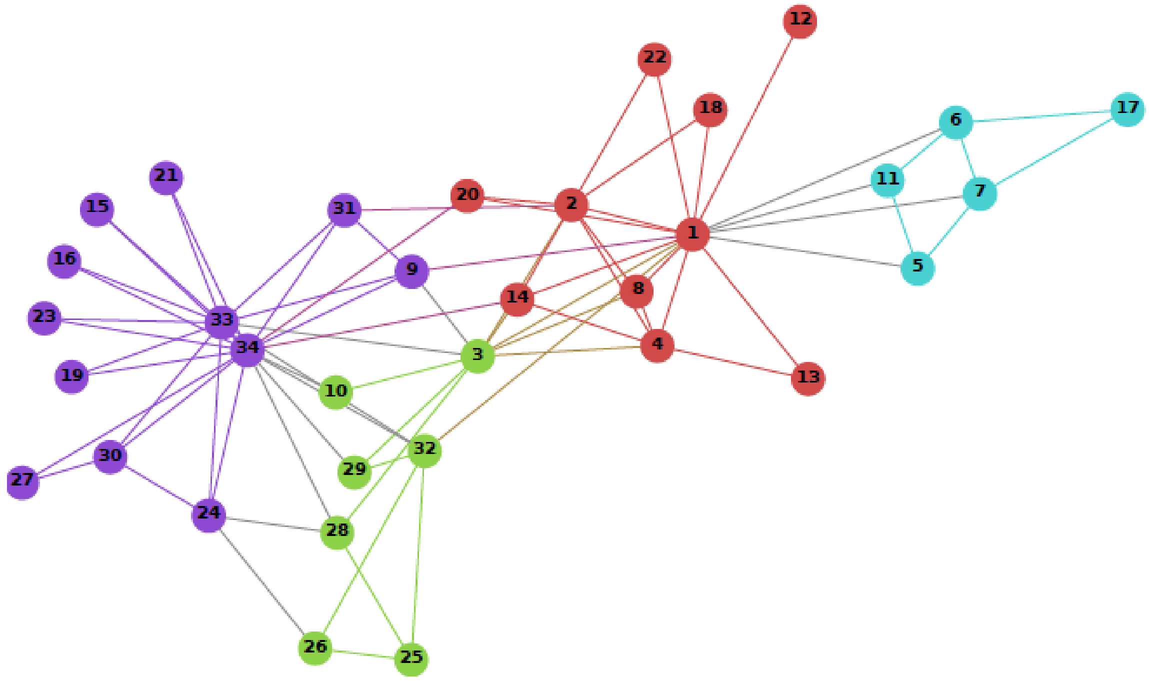
DeepWalk Online Learning of Social Representations

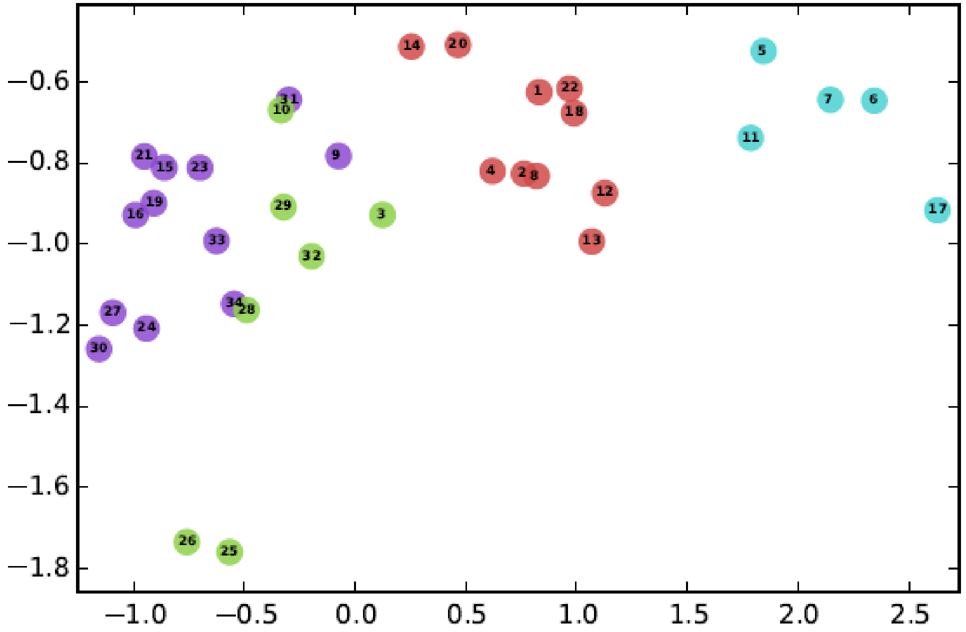
Bryan Perozzi, Rami Al-Rfou, Steven Skiena

Presenter: Xiangjue Dong

Introduction



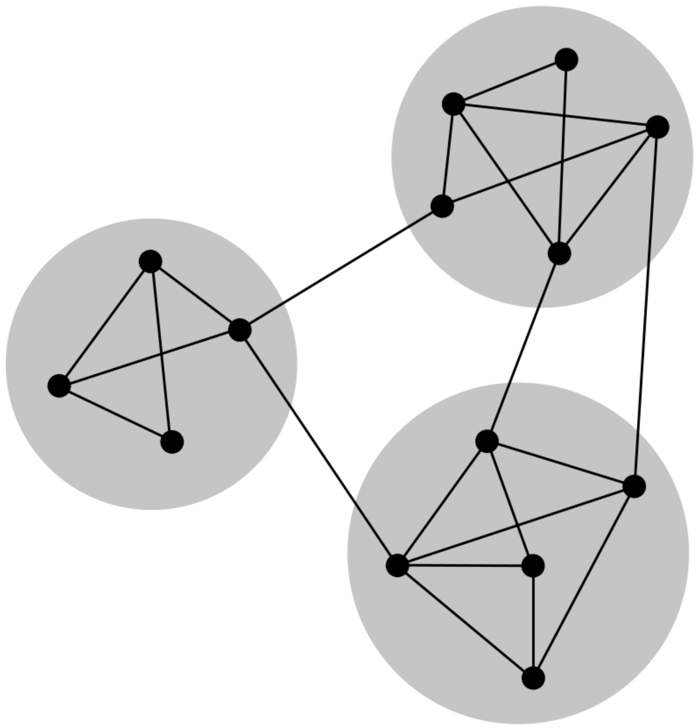
(a) Input: Karate Graph



(b) Output: Representation

Community Structure

- Pairs of nodes are both members of the same community(ies) - more likely to be connected
- Don’t share communities - less likely to be connected
- Zachary's Karate network:
  - A social network of a university karate club;
  - Captures 34 members of a karate club;
  - Documents links between pairs of members who interacted outside the club.



https://en.wikipedia.org/wiki/Community_structure
https://en.wikipedia.org/wiki/Zachary's_karate_club

Problem Definition

- $G = (V, E), E \in (V \times V)$
- $G_L = (V, E, X, Y)$

Social Representation Characteristics

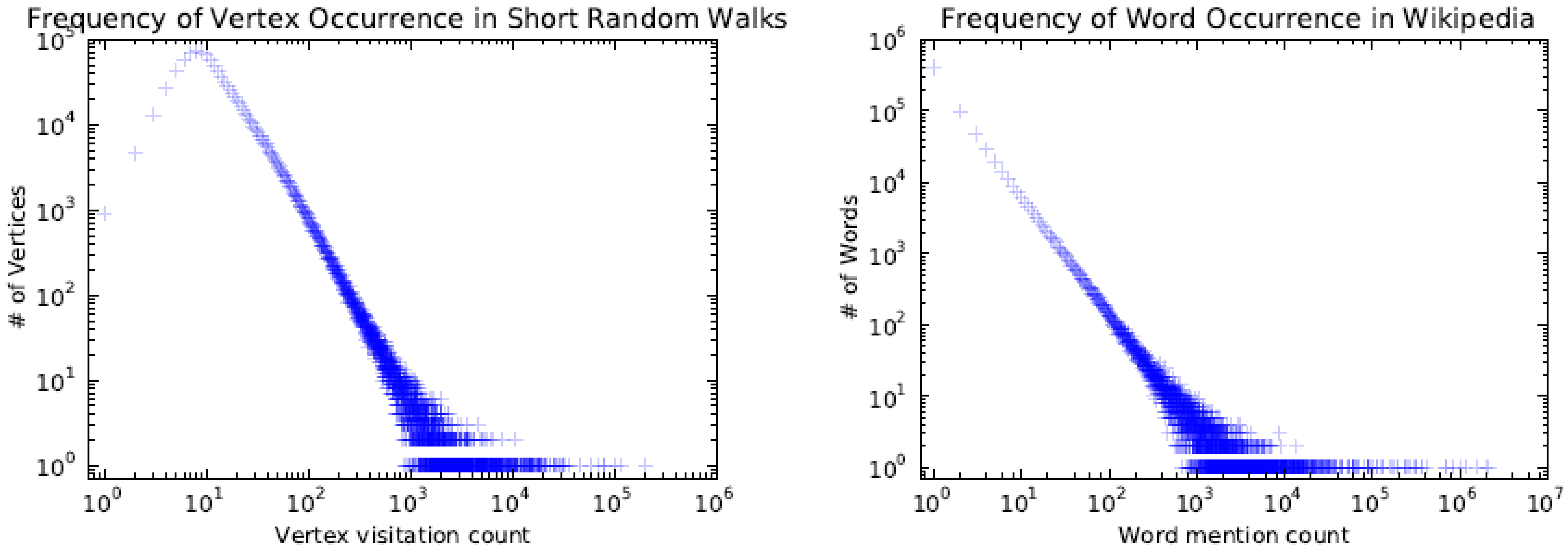
- Adaptability – evolve
- Community aware – represent similarity
- Low dimensional – generalize better, converge faster
- Continuous – more robust

Random Walks

- A stream of short random walks - extract information from a network.
- Two desirable properties:
  - Local exploration is easy to parallelize;
  - Accommodate small changes in the graph structure without the need for global recomputation.

Power Laws

- Scale-free Network:
  - A few nodes that are highly connected to other nodes in the network;
  - Its degree distribution follows a power law.



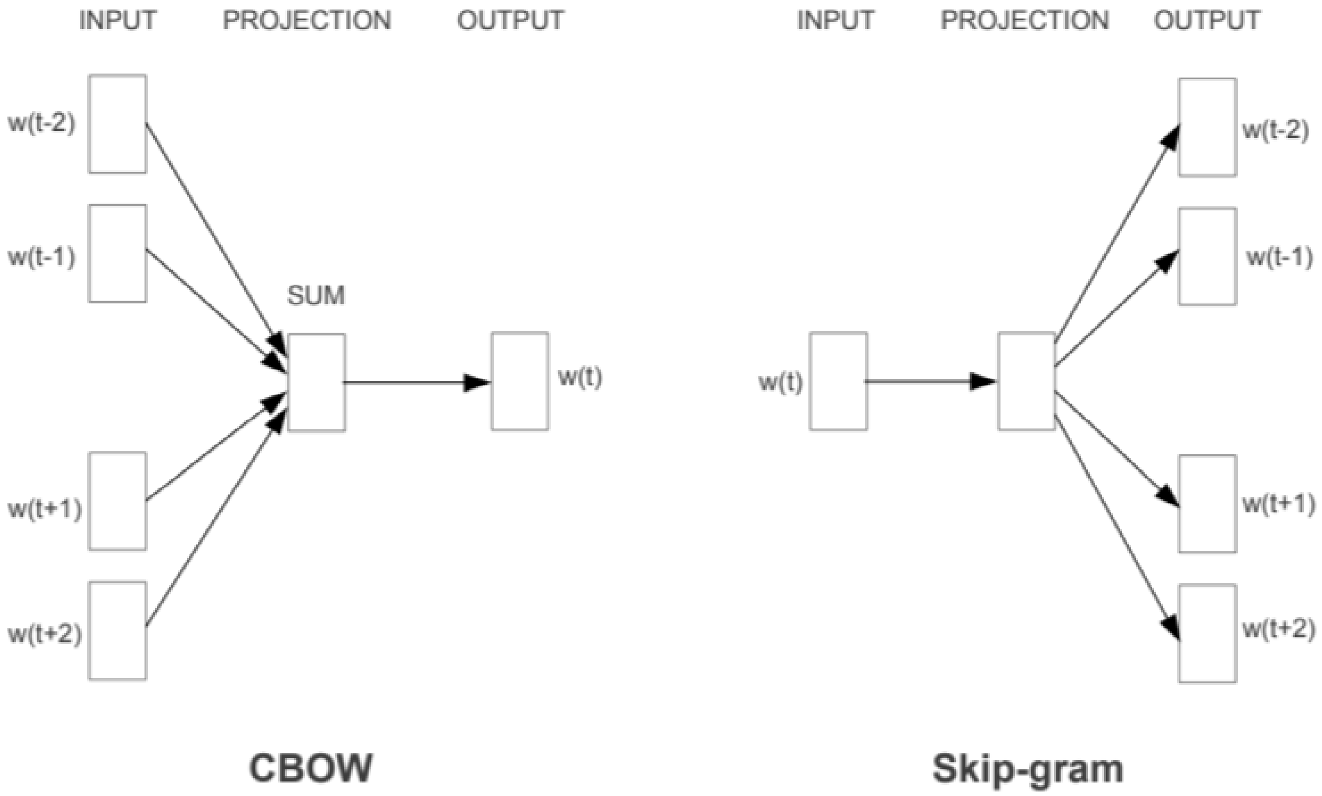
(a) YouTube Social Graph (b) Wikipedia Article Text

Language Modeling - Text

- Text
- Given $W_1^n = (w_0, w_1, \cdots, w_n)$
- $\max\ \Pr(w_n | w_0, w_1, \cdots, w_{n-1})$

Language Modeling - Text

- Text
- Given $W_1^n = (w_0, w_1, \cdots, w_n)$
- $\max \ \Pr(w_n | w_0, w_1, \cdots, w_{n-1})$
- CBOW
- Skip-gram



Language Modeling - Graph

- Text
- Given $W_1^n = (w_0, w_1, \cdots, w_n)$
- $\max \ \Pr(w_n | w_0, w_1, \cdots, w_{n-1})$
- CBOW
- Skip-gram

- Graph
- $\max \ \Pr(v_i | (v_1, v_2, \cdots, v_{i-1}))$

Language Modeling - Graph

- Text
- Given $W_1^n = (w_0, w_1, \cdots, w_n)$
- $\max \ \Pr(w_n | w_0, w_1, \cdots, w_{n-1})$
- CBOW
- Skip-gram

- Graph
- $\max \ \Pr(v_i | (v_1, v_2, \cdots, v_{i-1}))$
- $\Pr(v_i | (\emptyset(v_1), \emptyset(v_2), \cdots, \emptyset(v_{i-1})))$

Language Modeling - Graph

- Text
- Given $W_1^n = (w_0, w_1, \cdots, w_n)$
- $\max \ \Pr(w_n | w_0, w_1, \cdots, w_{n-1})$
- CBOW
- Skip-gram

- Graph
- $\max \ \Pr(v_i | (v_1, v_2, \cdots, v_{i-1}))$
- $\Pr(v_i | (\emptyset(v_1), \emptyset(v_2), \cdots, \emptyset(v_{i-1})))$
  - Problem:
    - Walk length ↑, computation unfeasible.

Language Modeling - Graph

- Text
- Given $W_1^n = (w_0, w_1, \cdots, w_n)$
- $\max \ \Pr(w_n | w_0, w_1, \cdots, w_{n-1})$
- CBOW
- Skip-gram

- Graph
- $\max \ \Pr(v_i | (v_1, v_2, \cdots, v_{i-1}))$
- $\Pr(v_i | (\emptyset(v_1), \emptyset(v_2), \cdots, \emptyset(v_{i-1})))$
  - Solution:
    - One word predicts context;
    - Context is composed of left and right side of the given word
    - Removes ordering constraint

Language Modeling - Graph

- Text
- Given $W_1^n = (w_0, w_1, \cdots, w_n)$
- $\max \ \Pr(w_n | w_0, w_1, \cdots, w_{n-1})$
- CBOW
- Skip-gram

- Graph
- $\Pr(v_i | (v_1, v_2, \cdots, v_{i-1}))$
- $\Pr(v_i | (\emptyset(v_1), \emptyset(v_2), \cdots, \emptyset(v_{i-1})))$
- $minimize_{\emptyset}$
- $-log\Pr(\{v_{i-w}, \cdots, v_{i-1}, v_{i+1}, \cdots, v_{i+w} | \emptyset(v_i))$
- Benefits:
    - Better captures “nearness”;
    - Speeds up training time.

Algorithm

- Random walk generator
- Update procedure.

Algorithm

Algorithm 1 DeepWalk(G, w, d, γ, t)

Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

```
Initialization: Sample Φ from U^{|V|×d}
Build a binary Tree T from V
for i = 0 to γ do
    O = Shuffle(V)
    for each v_i ∈ O do
        W_{v_i} = RandomWalk(G, v_i, t)
        SkipGram(Φ, W_{v_i}, w)
    end for
end for
```

Algorithm

Algorithm 1 DEEPWALK(G, w, d, γ, t)

Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t

Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
Build a binary Tree T from V
for $i = 0$ to γ **do**
    $\mathcal{O} = \text{Shuffle}(V)$
    for each $v_i \in \mathcal{O}$ **do**
        $\mathcal{W}_{v_i} = RandomWalk(G, v_i, \text{t})$
        SkipGram(Φ, $\mathcal{W}_{v_i}$, w)
    end for
end for

Algorithm

Algorithm 1 DEEPWALK(G, w, d, γ, t)

Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t

Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
Build a binary Tree T from V
for $i = 0$ to γ **do**
    $\mathcal{O} = \text{Shuffle}(V)$
    for each $v_i \in \mathcal{O}$ **do**
        $\mathcal{W}_{v_i} = RandomWalk(G, v_i,\text{t})$
        SkipGram(Φ, $\mathcal{W}_{v_i}$, w)
    end for
end for

Algorithm

Algorithm 1 DeepWalk(G, w, d, γ, t)

Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t

Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V|\times d}$

```
1: Initialization: Sample Φ from U^{|V|×d}
2: Build a binary Tree T from V
3: for i = 0 to γ do
4:     O = Shuffle(V)
5:     for each v_i ∈ O do
6:         W_{v_i} = RandomWalk(G, v_i, t)
7:         SkipGram(Φ, W_{v_i}, w)
8:     end for
9: end for
```

Algorithm

Algorithm 1 DeepWalk(G, w, d, γ, t)

Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t

Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V|\times d}$

```
Initialization: Sample Φ from U^{|V|×d}
Build a binary Tree T from V
for i = 0 to γ do
    O = Shuffle(V)
    for each v_i ∈ O do
        W_{v_i} = RandomWalk(G, v_i, t)
        SkipGram(Φ, W_{v_i}, w)
    end for
end for
```

Algorithm

Algorithm 1 DeepWalk(G, w, d, γ, t)

Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
Build a binary Tree T from V
for $i = 0$ to γ **do**
  $\mathcal{O} = \text{Shuffle}(V)$
  for each $v_i \in \mathcal{O}$ **do**
    $\mathcal{W}_{v_i} = RandomWalk(G, v_i, \text{t})$
    SkipGram(Φ, $\mathcal{W}_{v_i}$, w)
  end for
end for

Algorithm

Algorithm 2 SkipGram(Φ, $\mathcal{W}_{v_i}$, w)

```
for each v_j ∈ W_{v_i} do
    for each u_k ∈ W_{v_i}[j − w : j + w] do
        J(Φ) = − log Pr(u_k | Φ(v_j))
        Φ = Φ − α * ∂J/∂Φ
    end for
end for
```

$$J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$$

$$\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$$

Algorithm

Algorithm 2 SkipGram(Φ, $\mathcal{W}_{v_i}$, w)

for each $v_j \in \mathcal{W}_{v_i}$ **do**
    for each $u_k \in \mathcal{W}_{v_i}[j - w : j + w]$ **do**
        $J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$
        $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$
    end for
end for

Algorithm

Algorithm 2 SkipGram(Φ, $\mathcal{W}_{v_i}$, w)

1: **for each** $v_j \in \mathcal{W}_{v_i}$ **do**
2:     **for each** $u_k \in \mathcal{W}_{v_i}[j - w : j + w]$ **do**
3:         $J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$
4:         $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$
5:     **end for**
6: **end for**

Algorithm Variants

- Streaming
  - Implemented without entire graph
- Non-random walks

Experiments

- Datasets
  - BlogCatalog
  - Flickr
  - YouTube

Name	BlogCatalog	Flickr	YouTube
$\|V\|$	10,312	80,513	1,138,499
$\|E\|$	333,983	5,899,882	2,990,443
$\|\mathcal{Y}\|$	39	195	47
Labels	Interests	Groups	Groups

- Baseline methods
  - SpectralClustering – eigenvectors
  - Modularity – eigenvectors
  - EdgeCluster – k-means to cluster adjacency matrix
  - wvRN – relational classifier
  - Majority – most frequent labels

Results – BlogCatalog

	% Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1(%)	DeepWalk	**36.00**	**38.20**	**39.60**	**40.30**	**41.00**	**41.30**	41.50	41.50	42.00
	SpectralClustering	31.06	34.95	37.27	38.93	39.97	40.99	**41.66**	**42.42**	**42.62**
	EdgeCluster	27.94	30.76	31.85	32.99	34.12	35.00	34.63	35.99	36.29
	Modularity	27.35	30.74	31.77	32.97	34.09	36.13	36.08	37.23	38.18
	wvRN	19.51	24.34	25.62	28.82	30.37	31.81	32.19	33.33	34.28
	Majority	16.51	16.66	16.61	16.70	16.91	16.99	16.92	16.49	17.26
Macro-F1(%)	DeepWalk	**21.30**	**23.80**	25.30	26.30	27.30	27.60	27.90	28.20	28.90
	SpectralClustering	19.14	23.57	**25.97**	**27.46**	**28.31**	**29.46**	**30.13**	**31.38**	**31.78**
	EdgeCluster	16.16	19.16	20.48	22.00	23.00	23.64	23.82	24.61	24.92
	Modularity	17.36	20.00	20.80	21.85	22.65	23.41	23.89	24.20	24.97
	wvRN	6.25	10.13	11.64	14.24	15.86	17.18	17.98	18.86	19.57
	Majority	2.52	2.55	2.52	2.58	2.58	2.63	2.61	2.48	2.62

Table 2: Multi-label classification results in BlogCatalog

Results – Flickr

	% Labeled Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1(%)	DeepWalk	**32.4**	**34.6**	**35.9**	**36.7**	**37.2**	**37.7**	**38.1**	**38.3**	**38.5**	**38.7**
	SpectralClustering	27.43	30.11	31.63	32.69	33.31	33.95	34.46	34.81	35.14	35.41
	EdgeCluster	25.75	28.53	29.14	30.31	30.85	31.53	31.75	31.76	32.19	32.84
	Modularity	22.75	25.29	27.3	27.6	28.05	29.33	29.43	28.89	29.17	29.2
	wvRN	17.7	14.43	15.72	20.97	19.83	19.42	19.22	21.25	22.51	22.73
	Majority	16.34	16.31	16.34	16.46	16.65	16.44	16.38	16.62	16.67	16.71
Macro-F1(%)	DeepWalk	**14.0**	17.3	**19.6**	**21.1**	**22.1**	**22.9**	**23.6**	**24.1**	**24.6**	**25.0**
	SpectralClustering	13.84	**17.49**	19.44	20.75	21.60	22.36	23.01	23.36	23.82	24.05
	EdgeCluster	10.52	14.10	15.91	16.72	18.01	18.54	19.54	20.18	20.78	20.85
	Modularity	10.21	13.37	15.24	15.11	16.14	16.64	17.02	17.1	17.14	17.12
	wvRN	1.53	2.46	2.91	3.47	4.95	5.56	5.82	6.59	8.00	7.26
	Majority	0.45	0.44	0.45	0.46	0.47	0.44	0.45	0.47	0.47	0.47

Table 3: Multi-label classification results in Flickr

Results – YouTube

	% Labeled Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1(%)	DeepWalk	**37.95**	**39.28**	**40.08**	**40.78**	**41.32**	**41.72**	**42.12**	**42.48**	**42.78**	**43.05**
	SpectralClustering	—	—	—	—	—	—	—	—	—	—
	EdgeCluster	23.90	31.68	35.53	36.76	37.81	38.63	38.94	39.46	39.92	40.07
	Modularity	—	—	—	—	—	—	—	—	—	—
	wvRN	26.79	29.18	33.1	32.88	35.76	37.38	38.21	37.75	38.68	39.42
	Majority	24.90	24.84	25.25	25.23	25.22	25.33	25.31	25.34	25.38	25.38
Macro-F1(%)	DeepWalk	**29.22**	**31.83**	**33.06**	**33.90**	**34.35**	**34.66**	**34.96**	**35.22**	**35.42**	**35.67**
	SpectralClustering	—	—	—	—	—	—	—	—	—	—
	EdgeCluster	19.48	25.01	28.15	29.17	29.82	30.65	30.75	31.23	31.45	31.54
	Modularity	—	—	—	—	—	—	—	—	—	—
	wvRN	13.15	15.78	19.66	20.9	23.31	25.43	27.08	26.48	28.33	28.89
	Majority	6.12	5.86	6.21	6.1	6.07	6.19	6.17	6.16	6.18	6.19

Table 4: Multi-label classification results in YouTube

Reference

- Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD ’14). Association for Computing Machinery, New York, NY, USA, 701–710. DOI:https://doi.org/10.1145/2623330.2623732

Thank you