Fast and Accurate Network Embeddings via Very Sparse Random Projection

Haochen Chen[1], Syed Fahad Sultan[1], Yingtao Tian[1], Muhao Chen[2], Steven Skiena[1]

Stony Brook University[1]
University of California, Los Angeles[2]

One-sentence Summary

- *FastRP*: a network embedding method almost as performant as DeepWalk but runs 4,000 times faster

Outline

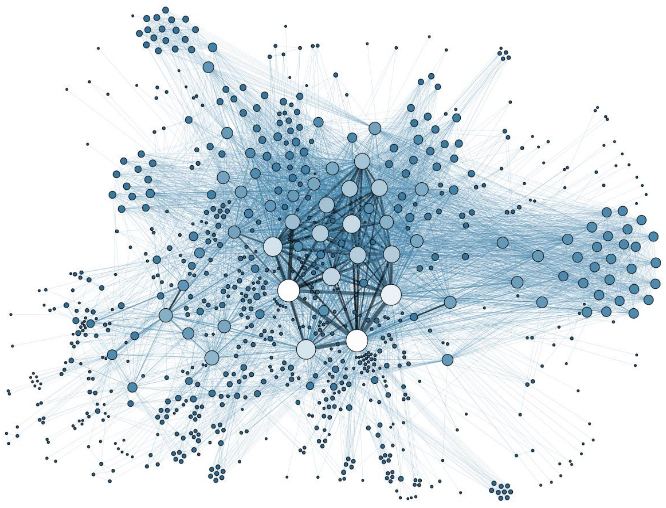
- Background and Motivation
- *FastRP* – Fast NE via Very Sparse Random Projection
- Evaluation

Outline

- Background and Motivation
- *FastRP* – Fast NE via Very Sparse Random Projection
- Evaluation

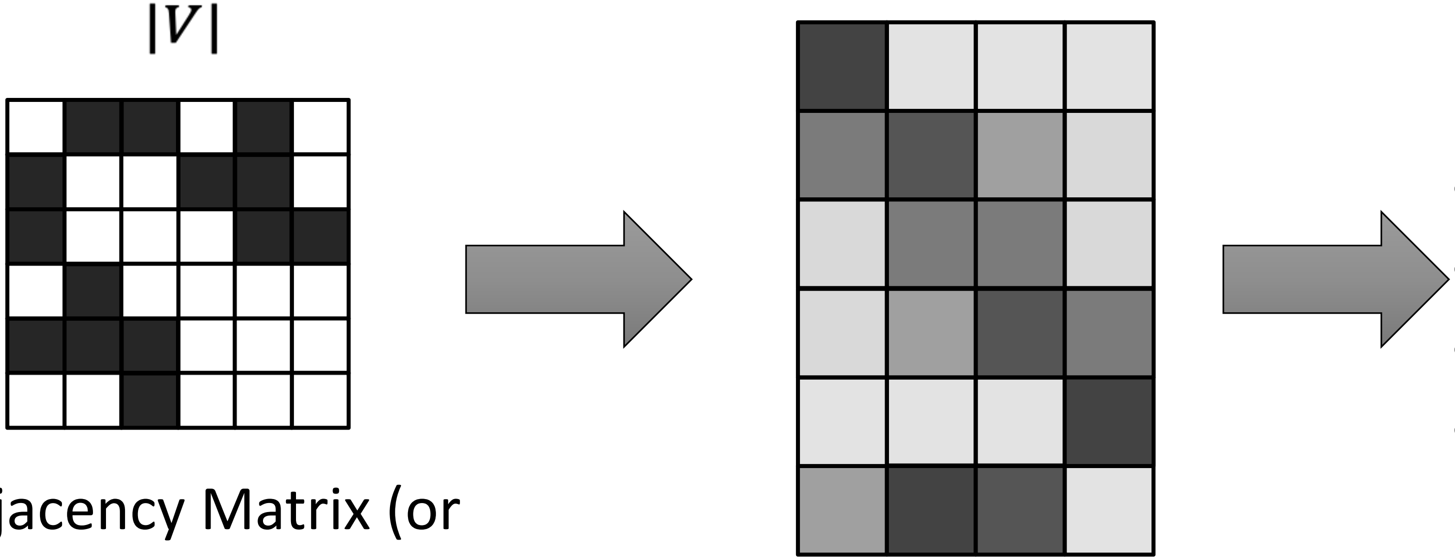
Background

- What is network embedding?
  - Low-dimensional latent representation of nodes in a network



$d \ll |V|$

$|V|$



- Node Classification
- Link Prediction
- Clustering
- …

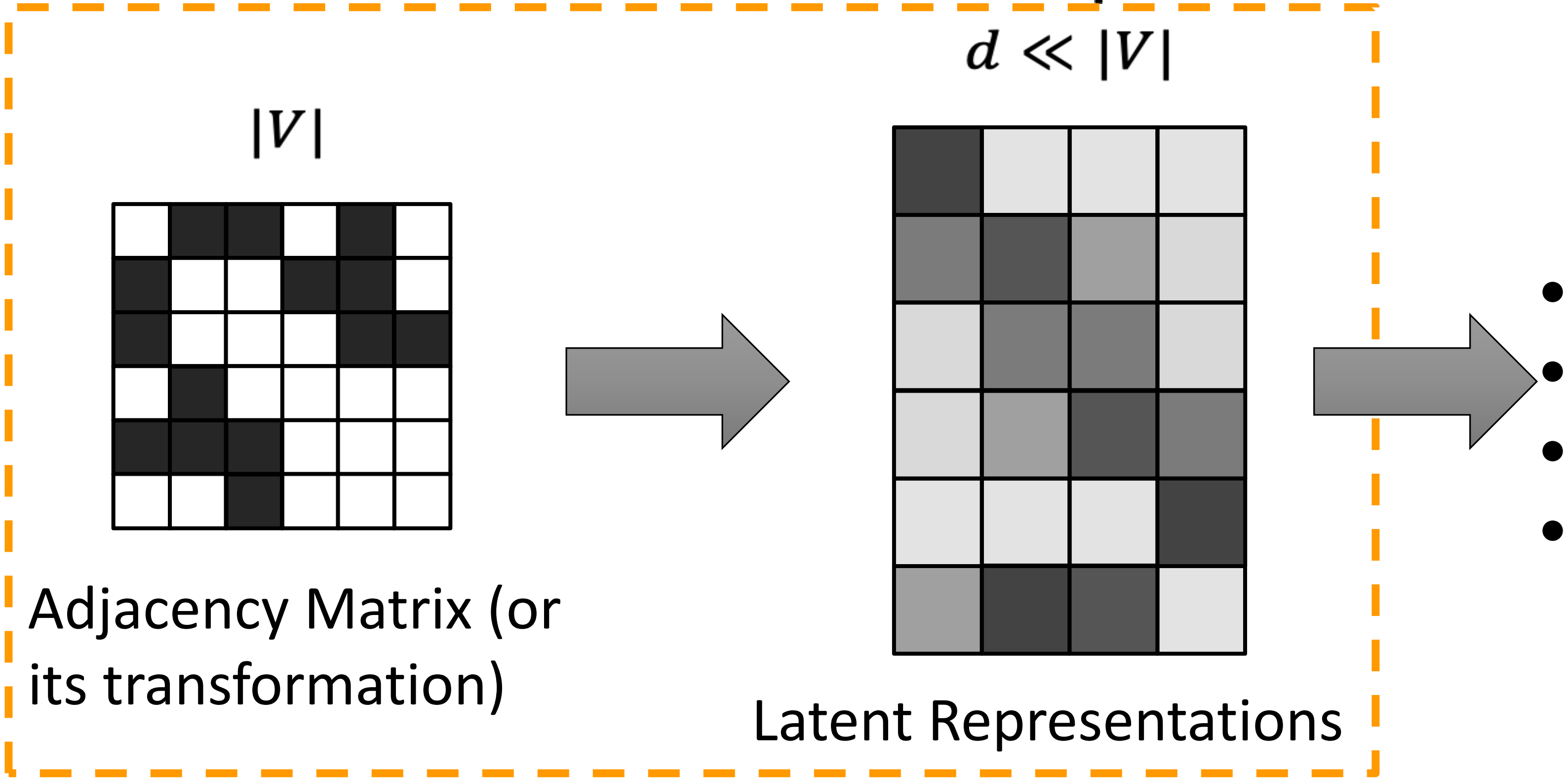
Adjacency Matrix (or its transformation)

Latent Representations

The Task of Network Embedding

Formally: given a network $G = (V, E)$

We aim to learn a mapping $\Phi : V \mapsto \mathbb{R}^{|V| \times d}$ from each vertex to a d-dimensional vector in the real space



- Node Classification
- Link Prediction
- Clustering
- …

Two Questions to Answer…

- What is an appropriate input matrix to use?
  - Transition matrix, the (normalized) Laplacian matrix, or the powers of these matrices
  - And sometimes we sample from an input matrix instead of explicitly construct it

1. Node Similarity Matrix Construction.

- What dimension reduction techniques should be applied on this input matrix?
  - Many choices: skip-gram, SVD, random projection, etc.

2. Dimension Reduction.

DeepWalk's Answer to the Two Questions

- Input matrix: Cor

$\boxed{\log}\left(\mathrm{v}\right.$

Algorithm 3: NetMF for a Small Window Size T

1. Compute $\boldsymbol{P}^1, \cdots, \boldsymbol{P}^T$;
2. Compute $\boldsymbol{M} = \frac{\mathrm{vol}(G)}{bT}\left(\sum_{r=1}^{T} \boldsymbol{P}^r\right)\boldsymbol{D}^{-1}$;
3. Compute $\boldsymbol{M}' = \max(\boldsymbol{M}, 1)$;
4. Rank-d approximation by SVD: $\log \boldsymbol{M}' = \boldsymbol{U}_d \boldsymbol{\Sigma}_d \boldsymbol{V}_d^\top$;
5. **return** $\boldsymbol{U}_d\sqrt{\boldsymbol{\Sigma}_d}$ as network embedding.

Element-wise transformation
(n^2 elements!)

Powers of the transition matrix
A

- (Note: direct computation of this matrix is not feasible, so DeepWalk samples from it instead)
- Dimension reduction: implicitly matrix factorization with Skip-gram

Motivation: Why is DeepWalk slow?

- DeepWalk samples node pairs from different powers of the transition matrix A, and then apply Skip-gram on these samples

- Drawback #1: huge number of samples needed
  - for a graph with 1M nodes, DeepWalk samples 80 random walks of length 40, and set the window size in Skip-gram to 10
  - Around 1M * 80 * 40 * 10 = 30B node pairs are sampled!

- Drawback #2: Skip-gram is not that fast
  - SGD-based optimization

Motivation: Scalability Issue with Existing Methods

- It takes about 30 days of CPU time to run DeepWalk on the Youtube graph (with 1M nodes)
  - Still, DeepWalk is one of the most scalable network embedding method
- Can we design a more scalable network embedding algorithm?

Motivation

We want a network embedding method that overcomes these drawbacks via:

- Design a proper input matrix with lessons learn from **DeepWalk**:
  - Raise A to higher power
  - Proper transformation / normalization of matrix elements
- Employ a scalable dimension reduction technique

Outline

- Background and Motivation
- *FastRP* – Fast NE via Very Sparse Random Projection
- Evaluation

Very Sparse Random Projection: Scalable Dimension Reduction

- Random projection: optimization-free method for dimension reduction
- Given a input similarity matrix $S \in \mathbb{R}^{N \times N}$, simply compute $U = S \cdot R$, $R \in \mathbb{R}^{N \times d}$ as the embedding matrix where each element of R is sampled from the following distribution:

$$\mathbf{R}_{ij} = \begin{cases} \sqrt{s} \text{ with probability } \dfrac{1}{2s} \\ 0 \text{ with probability } 1 - \dfrac{1}{s} \\ -\sqrt{s} \text{ with probability } \dfrac{1}{2s} \end{cases}$$

Santosh S Vempala. 2005. The random projection method. Vol. 65. American Mathematical Soc.

Random Projection: Input Similarity Matrix

- Similar to DeepWalk, we can take a weighted combination of different powers of A:
  - $S = \alpha_0 I + \alpha_1 A + \alpha_2 A^2 + \cdots + \alpha_k A^k$
  - $k = 5$ is usually enough

- But we are missing the element-wise log transformation part and we want to avoid that
  - But why is the transformation important anyway?

Why is element-wise transformation important?

- Intuitive explanation:

$$\mathbf{A}_{ij}^{k} \to d_j/2m \quad \text{when} \quad \boldsymbol{k} \to \infty$$

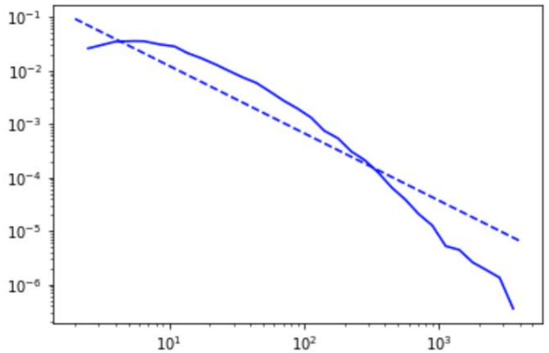
- The distribution of entries in A follows the node degree distribution of G
    - which is usually skewed，不是对称的（幂律分布）
- The element-wise log-transformation reduces data skewness

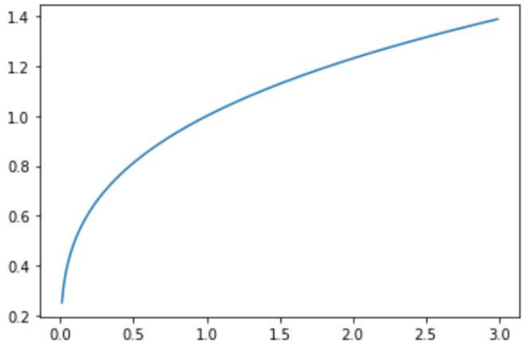
Our alternative element-wise transformation

- Leverage the fact that $\mathbf{A}_{ij}^{k} \to d_j/2m$
- We take $\tilde{\mathbf{A}}_{ij}^{k} = \mathbf{A}_{ij}^{k} \cdot \left(\frac{d_j}{2m}\right)^{\lambda-1} \approx \mathbf{A}_{ij}^{k} \cdot (\mathbf{A}_{ij}^{k})^{\lambda-1} \approx (\mathbf{A}_{ij}^{k})^{\lambda}$

- This is similar to the idea of Tukey transformation
- λ is a tunable parameter between 0 and 1
- λ=0.5: taking squared root of features

$$\tilde{\mathbf{A}}^k = \mathbf{A}^k \cdot \mathbf{L} \text{ where } \mathbf{L} = \mathrm{diag}\left(\left(\frac{d_1}{2m}\right)^{\beta}, \ldots, \left(\frac{d_n}{2m}\right)^{\beta}\right).$$





Our Algorithm

- Given an input transition matrix A, simply compute:

$$\mathbf{N} = \left(\alpha_1 \tilde{\mathbf{A}} + \alpha_2 \tilde{\mathbf{A}}^2 + \ldots + \alpha_k \tilde{\mathbf{A}}^k\right) \cdot \mathbf{R}$$

  Where R is a sparse random matrix.

- Essentially, we only need to compute $\tilde{\mathbf{A}} \cdot \mathbf{R}, \tilde{\mathbf{A}}^{\mathbf{2}} \cdot \mathbf{R}, \cdots, \tilde{\mathbf{A}}^{\mathbf{k}} \cdot \mathbf{R}$

Our Algorithm

- But computing $\tilde{A}^k$ is still slow – can we avoid it?
- Yes!

$$\tilde{A}^2 \cdot R = A \cdot \left(\tilde{A} \cdot R\right)$$
$$\tilde{A}^3 \cdot R = A \cdot \left(\tilde{A}^2 \cdot R\right)$$
$$\cdots$$
$$\tilde{A}^k \cdot R = A \cdot \left(\tilde{A}^{k-1} \cdot R\right)$$

- $R, \tilde{A} \cdot R, \tilde{A}^2 \cdot R, \cdots$ are all $N \times d$ matrix
- We always multiply an $N \times d$ matrix with a very sparse $N \times N$ matrix A

Our Algorithm

Algorithm 1 FastRP($\mathbf{A}$)

Input:

graph transition matrix $\mathbf{A}$, embedding dimensionality d, maximum power k, normalization strength β, weights $\alpha_1, \alpha_2, \ldots, \alpha_k$

Output: matrix of node representations $\mathbf{N} \in \mathbb{R}^{n \times d}$

1: Produce $\mathbf{R} \in \mathbb{R}^{n \times d}$ according to Eq. 6
2: $\mathbf{N}_1 \leftarrow \mathbf{A} \cdot \mathbf{L} \cdot \mathbf{R}$ where $\mathbf{L}_{ij} = \left(\frac{d_j}{2m}\right)^{\beta}$
3: **for** $i = 2$ to n **do**
4: $\quad \mathbf{N}_i \leftarrow \mathbf{A} \cdot \mathbf{N}_{i-1}$
5: **end for**
6: $\mathbf{N} = \alpha_1 \mathbf{N}_1 + \ldots + \alpha_k \mathbf{N}_k$
7: **return N**

Time complexity: $O((n+m) \cdot k \cdot d)$

Outline

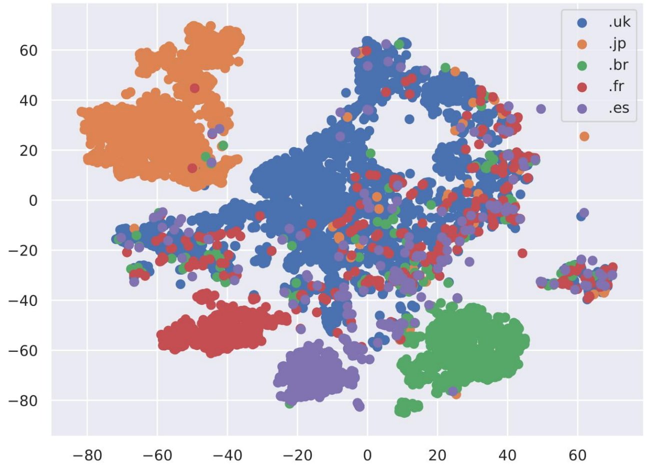
- Background and Motivation
- *FastRP* – Fast NE via Very Sparse Random Projection
- Evaluation

CPU Time Comparison

Dataset	**Algorithm**					Speedup over
	FastRP	RandNE	LINE	DeepWalk	node2vec	DeepWalk
WWW-200K	136.0 seconds	169.8 seconds	4.6 hours	6.9 days	63.8 days	**4383x**
WWW-10K	7.8 seconds	13.6 seconds	3.2 hours	9.2 hours	59.8 hours	**4246x**
Blogcatalog	6.0 seconds	10.5 seconds	3.0 hours	8.7 hours	41.2 hours	**5220x**
Flickr	33.1 seconds	45.1 seconds	4.2 hours	3.1 days	28.5 days	**8091x**

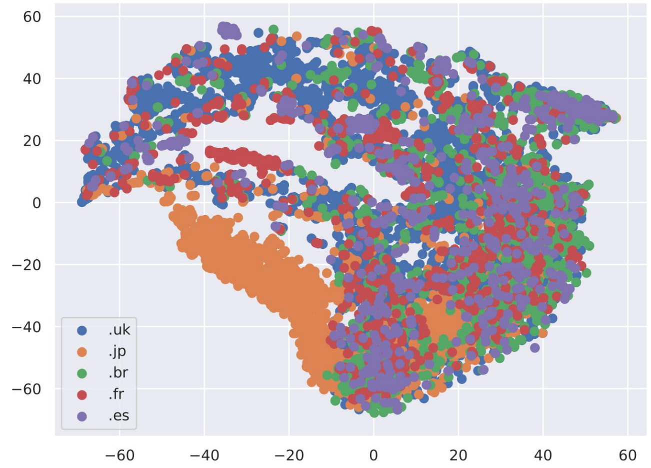
- 4000x faster than DeepWalk

Visualization



FastRP

DeepWalk



RandNE

KNN from Node Embeddings

Methods	FastRP	DeepWalk	RandNE	FastRP	DeepWalk	RandNE
Websites		nytimes.com			delta.com	
Neighbors	huffingtonpost.com washingtonpost.com cnn.com npr.org latimes.com	washingtonpost.com huffingtonpost.com cnn.com cbsnews.com time.com	huffingtonpost.com washingtonpost.com forbes.com cnn.com npr.org	aa.com united.com usairways.com alaskaair.com jetblue.com	aa.com united.com usairways.com southwest.com jetblue.com	aa.com southwest.com united.com expedia.com priceline.com
Methods	FastRP	DeepWalk	RandNE	FastRP	DeepWalk	RandNE
Websites		vldb.org			arsenal.com	
Neighbors	sigmod.org comp.nus.edu.sg sigops.org cidrdb.org cse.iitb.ac.in	sigmod.org morganclaypool.com kdd.org doi.acm.org informatic.uni-trier.de	comp.nus.edu.sg cs.sfu.ca cs.rpi.edu nlp.stanford.edu theory.stanford.edu	chelseafc.com mcfc.co.uk nufc.co.uk avfc.co.uk tottenhamhotspur.com	chelseafc.com tottenhamhotspur.com manutd.com mcfc.co.uk thefa.com	liverpoolfc.com manutd.com chelseafc.com skysports.com tottenhamhotspur.com

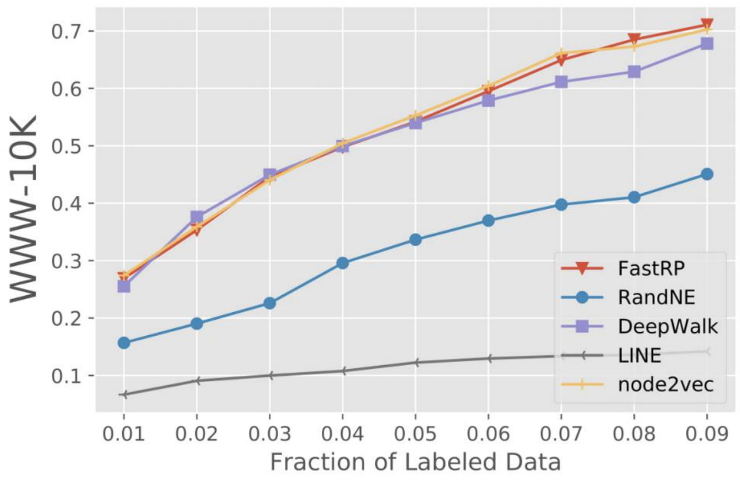
Multi-label Node Classification

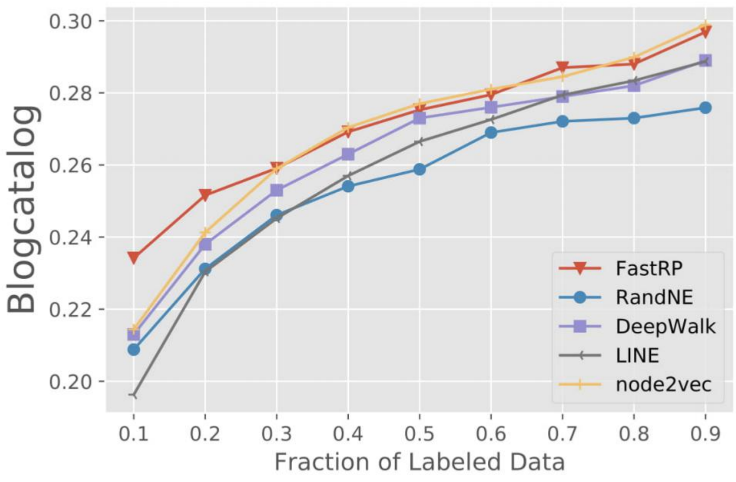
Algorithm	Dataset		
	WWW-10K	BlogCatalog	Flickr
LINE	6.66	19.63	10.69
node2vec	**27.42**	21.44	11.89
DeepWalk	25.54	21.30	14.00
RandNE	15.68	20.88	13.64
FastRP	26.92	**23.43**	**15.02**

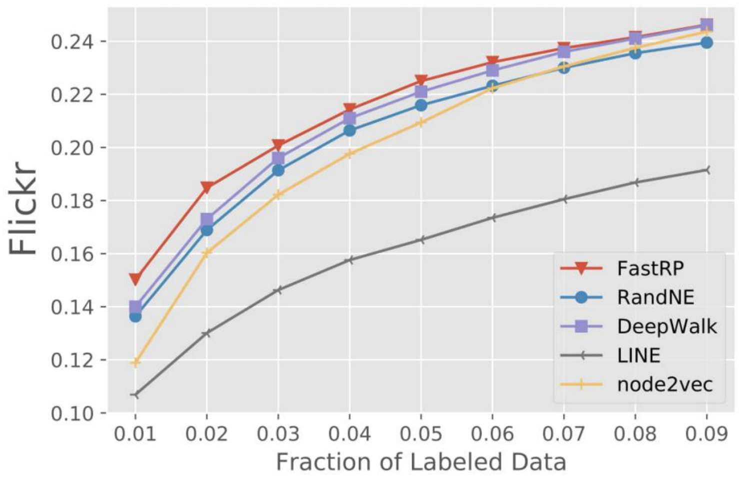
Table 4: Macro F_1 scores of all methods on WWW-10K, BlogCatalog, and Flickr in percentage (Section 4.6).

- FastRP has best or almost the best performance

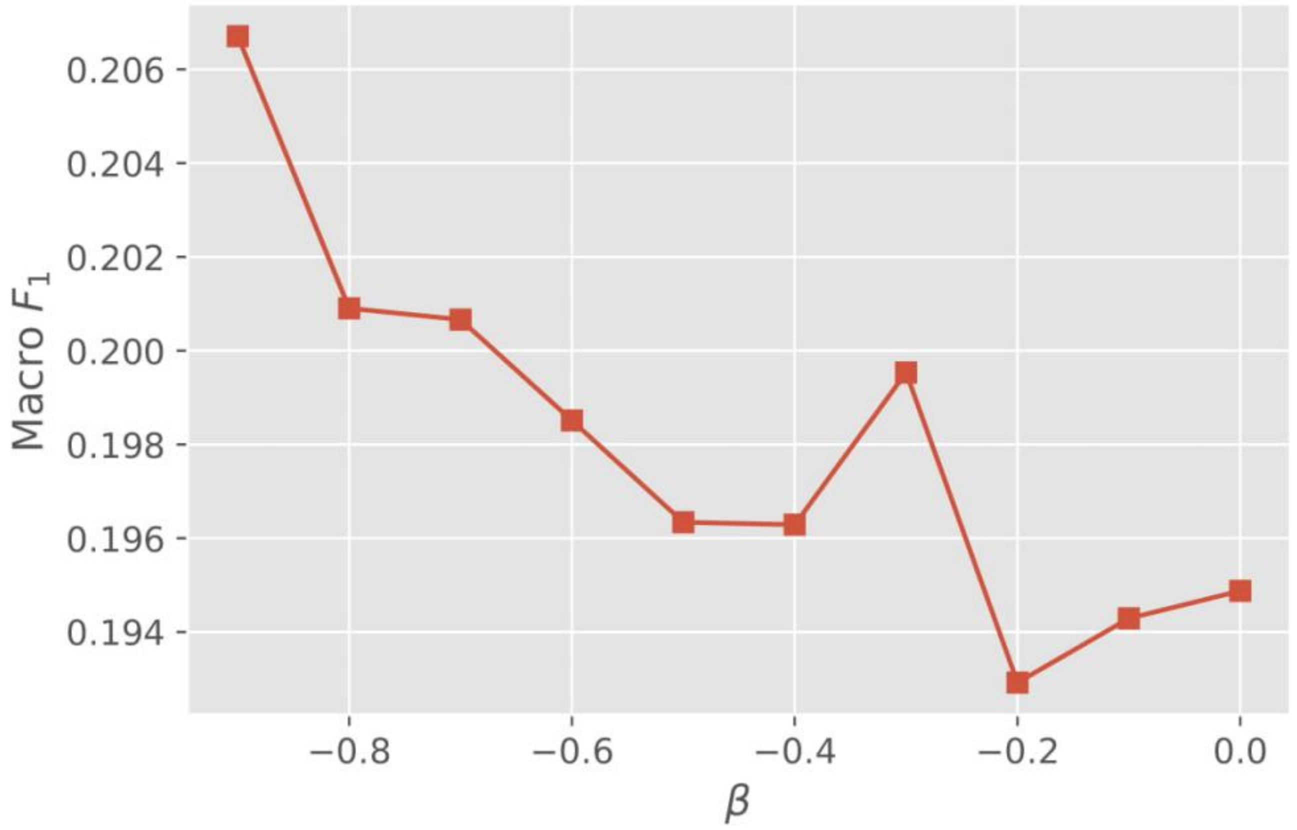
Detailed Node Classification Result







Normalization is Important



- $\boldsymbol{\beta = \lambda - 1}$
- $\boldsymbol{\beta = 0}$: no normalization

Thanks!

- Code is available at: http://bit.ly/fastrp-cikm
- For questions, feel free to contact me at haocchen@cs.stonybrook.edu
- Check out our poster tonight!