Dynamic Embedding on Textual Networks via a Gaussian Process

Presenter: Pengyu Cheng

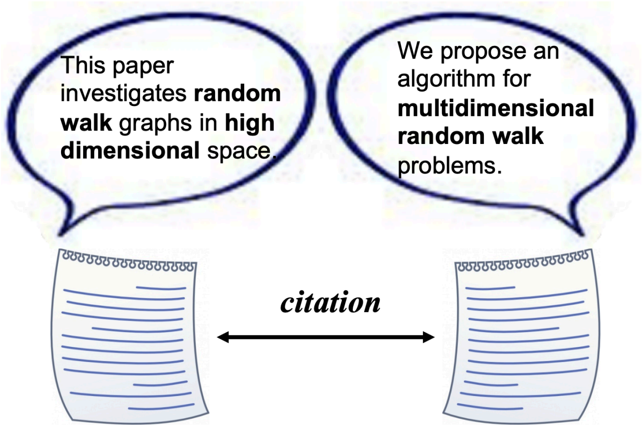
Joint work with: Yitong Li, Xinyuan Zhang, Liqun Chen, David Carlson, Lawrence Carin

Duke University

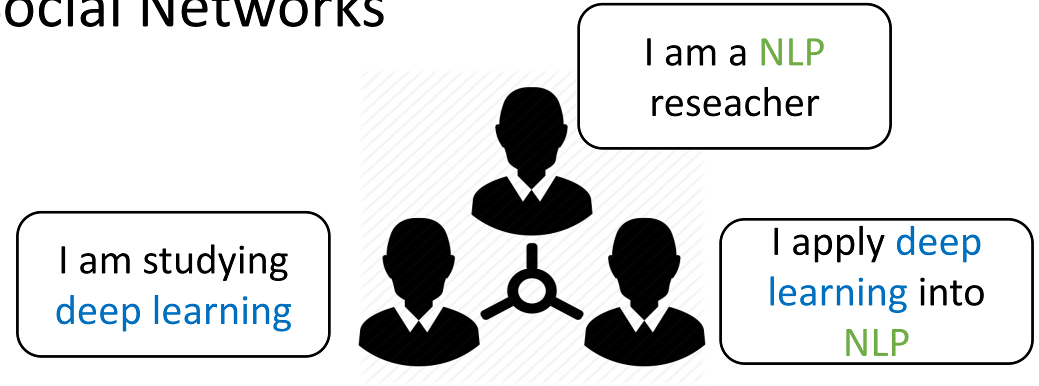
Textual Networks

- Networks with textual information as attributes from each node.

- Examples:
  - Citation Networks



  - Social Networks



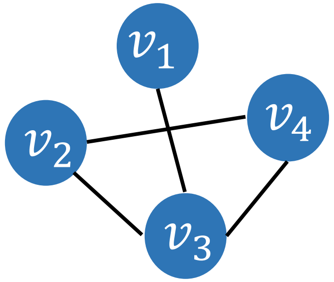
- Tasks on textual networks:
  - Link Prediction
  - Node Classification
  - Etc.

Textual Network Embedding

- Textual Embedding:

$$\rightarrow \left[\, \boldsymbol{X} \,\right]$$

- Network (Structural) Embedding:



$$\rightarrow \left[\, \boldsymbol{S} \,\right]$$

- Textual Network Embedding:

$$\boldsymbol{H} = [\boldsymbol{X}; \boldsymbol{S}]$$

Dynamic Textual Network Embedding

- Textual Networks are always changing:
- For text:
  - Textual information can be modified, e.g., user profile changes
- For network connection:
  - Unseen nodes come, e.g., new user registers.
  - Connection changes, e.g., people begin or lose relationships.
- How to speedily update the nework embeddings when network changes?

Objective

- We aim to update the embeddings when networks change,

  without re-training the whole model.

- For textual embedding:

  Inductively learn a universal text encoder.

- For structural embedding:

  Previous methods require re-train all the structural embeddings.

- We address this problem from a Bayesian Deep Learning perspective.

Main Idea

- We treat structural embeddings in the same network as samples from the same distribution.
- We provide a Gaussian prior to the structural embeddings
- When network changes, we dynamically update the structural embeddings by calculating the posterior distribution via the Bayes's rule.

Gaussian Process

- Define latent signal function $f(\boldsymbol{x}) \sim \mathrm{GP}(\mathbf{0}, k(\boldsymbol{x}, \boldsymbol{x}'))$
  over textual embedding $\boldsymbol{x}$, $k(\cdot, \cdot)$ is a kernel function.

$$[f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_n)] \sim \mathcal{N}(\mathbf{0}, [k(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{n \times n})$$

- Propagate $f(\boldsymbol{x})$ on the network by random walk.

- We use the propagated signals as the prior of structural embeddings.

$$[\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n] \sim \mathcal{N}(\mathbf{0}, \boldsymbol{P}^T K_{xx} \boldsymbol{P})$$

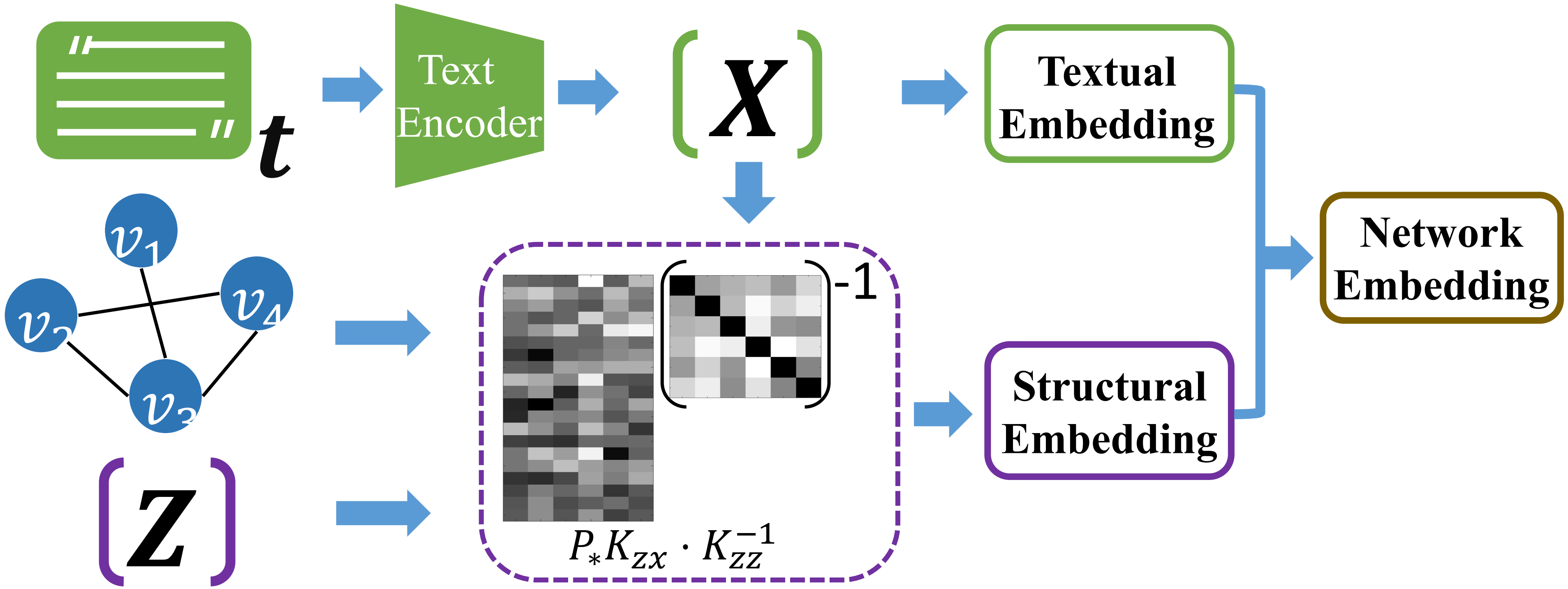
$$K_{xx} = [k(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{n \times n}$$

Inducing points

- Gaussian Process is computationally complex with large data size.
- We learn inducing points on graph to reduce the complexity.
- With learned inducing point $\boldsymbol{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m]$ and corresponding pseudo-structural embeddings $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m]$

We can infer posterior of the structural embeddings $p(\boldsymbol{S}|\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{U})$ by the Bayes's rule.

Framework



t
Text Encoder
X
Textual Embedding
v_1
v_2
v_3
v_4
Z
-1
$P_* K_{zx} \cdot K_{zz}^{-1}$
Structural Embedding
Network Embedding

Negative Sampling Loss Function

- Obtain the textual network embedding $\boldsymbol{H} = [\boldsymbol{X}; \boldsymbol{S}]$,
- We train the whole model with unsupervised negative sampling:

$$\mathcal{L} = -\frac{1}{|\mathcal{E}|} \sum_{(v_i, v_j) \in \mathcal{E}} \log\left(\sigma(\boldsymbol{h}_i^\intercal \boldsymbol{h}_j)\right) - \frac{1}{N_s} \sum_{(v_i, v_k) \notin \mathcal{E}} \log\left[1 - \sigma(\boldsymbol{h}_i^\intercal \boldsymbol{h}_k)\right]$$

Experiments: Static Networks

- Link prediction

	Cora				
%Training Edges	**15%**	**35%**	**55%**	**75%**	**95%**
MMB (Airoldi et al. 2008)	54.7	59.5	64.9	71.1	75.9
node2vec (Grover and Leskovec 2016)	55.9	66.1	78.7	85.9	88.2
LINE (Tang et al. 2015)	55.0	66.4	77.6	85.6	89.3
DeepWalk (Perozzi, Al-Rfou, and Skiena 2014)	56.0	70.2	80.1	85.3	90.3
TADW (Yang et al. 2015)	86.6	90.2	90.0	91.0	92.7
CANE (Tu et al. 2017)	86.8	92.2	94.6	95.6	97.7
DMATE (Zhang et al. 2018a)	91.3	93.7	96.0	97.4	98.8
WANE (Shen et al. 2019)	91.7	94.1	96.2	**97.5**	**99.1**
DetGP (Wavg) only Text	83.4	89.1	89.9	90.9	92.3
DetGP (Wavg) only Struct	85.4	89.7	91.0	92.7	94.1
DetGP (Wavg)	92.8	94.8	95.5	96.2	97.5
DetGP (DWavg)	**93.4**	**95.2**	**96.3**	**97.5**	98.8

Experiments: Static Network

- Node classification:

	Cora				DBLP			
%Training Nodes	**10%**	**30%**	**50%**	**70%**	**10%**	**30%**	**50%**	**70%**
LINE(Tang et al. 2015)	53.9	56.7	58.8	60.1	42.7	43.8	43.8	43.9
TADW (Yang et al. 2015)	71.0	71.4	75.9	77.2	67.6	68.9	69.2	69.5
CANE (Tu et al. 2017)	81.6	82.8	85.2	86.3	71.8	73.6	74.7	75.2
DMTE (Zhang et al. 2018a)	81.8	83.9	86.3	87.9	72.9	74.3	75.5	76.1
WANE (Shen et al. 2019)	81.9	83.9	86.4	88.1	NA	NA	NA	NA
DetGP (Wavg) only Text	78.1	81.2	84.7	85.3	71.4	73.3	74.2	74.9
DetGP (Wavg) only Struct	70.9	79.7	81.5	82.3	70.0	71.4	72.6	73.3
DetGP (Wavg)	80.5	85.4	86.7	88.5	76.9	78.3	79.1	79.3
DetGP (DWavg)	**83.1**	**87.2**	**88.2**	**89.8**	**78.0**	**79.3**	**79.6**	**79.8**

Experiments: Dynamic Networks

- Link prediction

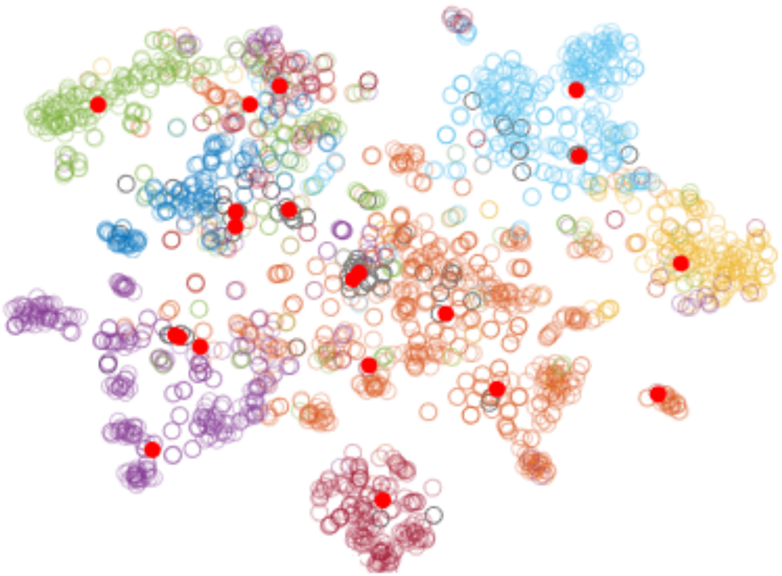
	Cora				HepTh			
%Training Nodes	**10%**	**30%**	**50%**	**70%**	**10%**	**30%**	**50%**	**70%**
Only Text (Wavg)	61.2	77.9	87.9	90.3	68.3	83.7	84.2	86.9
Neighbor-Aggregate (Max-Pooling)	54.6	69.1	78.7	87.3	59.6	78.3	79.9	80.7
Neighbor-Aggregate (Mean)	61.8	78.4	88.0	91.2	68.2	83.9	85.5	88.3
GraphSAGE (Max-Pooling)	62.1	78.6	88.6	92.4	68.4	85.8	88.1	91.2
GraphSAGE (Mean)	62.2	79.1	88.9	92.6	69.1	85.9	89.0	92.4
DetGP	**62.9**	**81.1**	**90.9**	**93.0**	**70.7**	**86.6**	**90.7**	**93.3**

- Node Classification

	Cora				DBLP			
% Training Nodes	**10%**	**30%**	**50%**	**70%**	**10%**	**30%**	**50%**	**70%**
Only Text (Wavg)	60.2	76.3	83.5	84.8	56.7	67.9	70.4	73.5
Neighbor-Aggregate (Max-Pooling)	55.8	70.2	78.4	80.5	51.8	60.5	68.3	70.6
Neighbor-Aggregate (Mean)	60.1	77.2	84.1	85.0	56.8	68.2	71.3	74.7
GraphSAGE (Max-Pooling)	61.3	78.2	85.1	86.3	58.9	69.1	72.4	74.9
GraphSAGE (Mean)	61.4	78.4	85.5	**86.6**	59.0	69.3	72.7	75.1
DetGP	**62.1**	**79.3**	**85.8**	**86.6**	**60.2**	**70.1**	**73.2**	**75.8**

Experiments:

- T-SNE visualization of learned
  structural embeddings on Cora dataset.



Source Code: https://github.com/Linear95/DetGP

Thank you!