Link Prediction on the Patent Citation Network

Samuel Chen[1] **David Dang**[1] **Robert Macy**[2] **Chris Rockwell**[2]

[1] *Department of Aerospace Engineering* [2] *Electrical Engineering and Computer Science*

EECS 598-008, W19
Advanced Data Mining
EECS

Problem Definition

- Link prediction (LP) is cast as a binary classification problem
  - Does a link exist between any two pair of nodes?
- **INPUT**: Adjacency list
- **OUTPUT**: Ranked list of most likely edges

Motivation

- Generate citation recommendations on new patents
- LP on the Patent Citation Dataset is unprecedented
  - LP on large and temporal graphs is less well studied
- Compare the performance of SDNE to other classical network embedding methods on different graph types

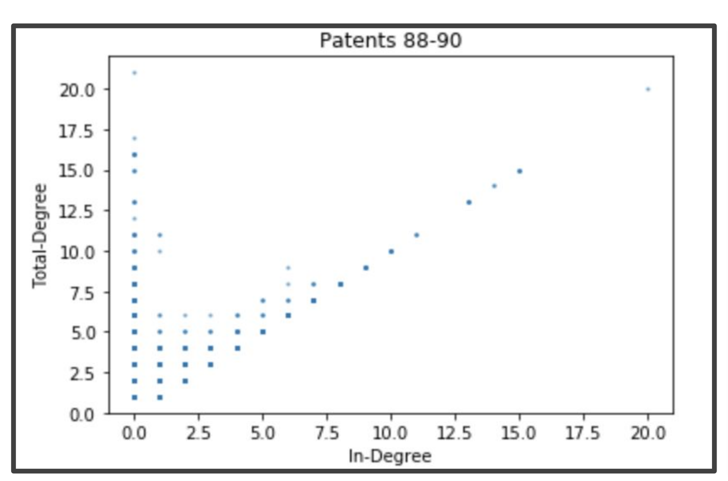
Datasets

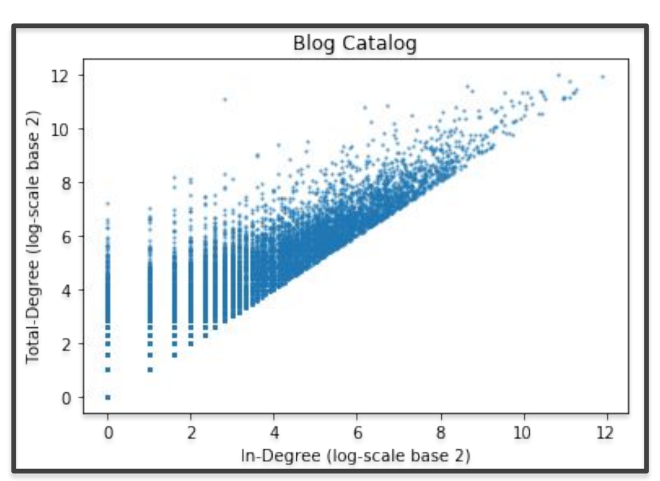
Patent Citation Dataset

- *1988 - 1989 subgraph:* 40K nodes, 30K edges
- *1990 - 1996 subgraph:* 580K nodes, 1.2M edges
  - Yielded poor results because of size
- *"Future" test set:* 500 proceeding nodes with 2 or more outward edges, inward edges removed

Blog Catalog: 10K nodes, 300K edges

- Undirected graph





References

1. Grover and Leskovec. node2vec: Scalable Feature Learning for Networks. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016.
2. Rodriguez et al. Graph mining algorithms for the analysis of patent citation networks, *Dissertation, Rutgers University.*
3. Wang et al. Structural deep network embedding. *In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM 2016, 1225–1234.
4. Niwattanakul et al. Using of Jaccard coefficient for keywords similarity. In Proceedings of the international multiconference of engineers and computer scientists 2013 (Vol. 1, No. 6, pp. 380-384).
5. Lerer et al. (2019). PyTorch-BigGraph: A Large-scale Graph Embedding System. arXiv preprint 2019, arXiv:1903.12287.

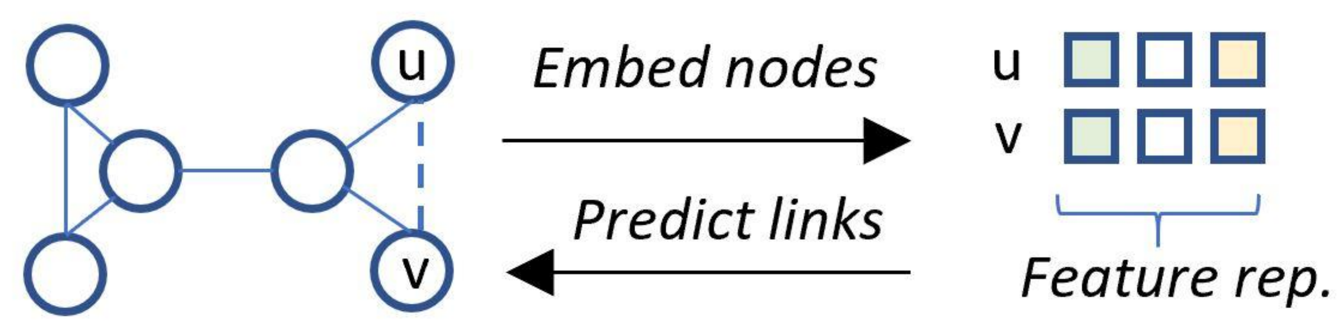
Our Approach

Data Partitioning

- Sub-graphs are divided into training and test sets
- 15% of the links in the sub-graph are hidden from the training set

Link Prediction from Node Embeddings

- Methods embed similar nodes close together
- Use node embeddings to predict hidden links



Approach 1: SDNE

Nodes embedded by optimizing weighted loss from:

- Supervised "first-order proximity"
  - Influenced by Leplacian Eigenmaps
  - Captures pairwise similarities i.e. common neighbors
- Unsupervised autoencoder "second-order proximity"
  - Mimics Graph Convolutional Network
  - Captures global structure i.e. role in network
- High unsupervised weight should help in sparse graphs

Approach 2: node2vec

- Focused on learning low-dimensional representation learning
- Node embeddings generated through random walk approach
- Random walks can be biased
  - Uses unbiased in this work

Approach 3: Jaccard Coefficient

- Predicts whether two nodes share a relationship, in this case a link, based on the intersection over union of their neighborhoods
- No learning involved, but very expensive at inference time

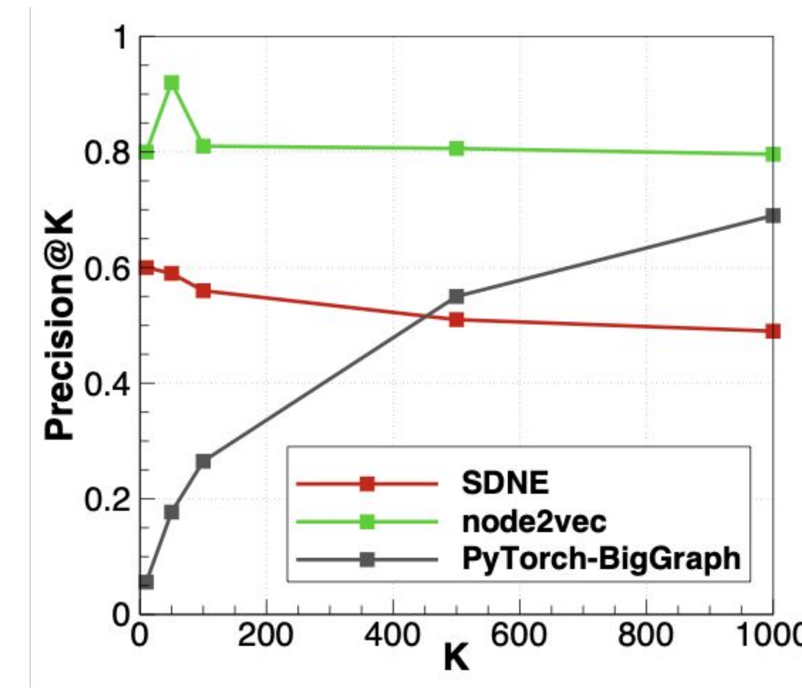
Approach 4: PyTorch BigGraph

- Learns node embedding by minimizing distance between adjacent nodes
- Gains efficiency by augmenting negative sampling with uniform samples used as negatives

Experimental Results

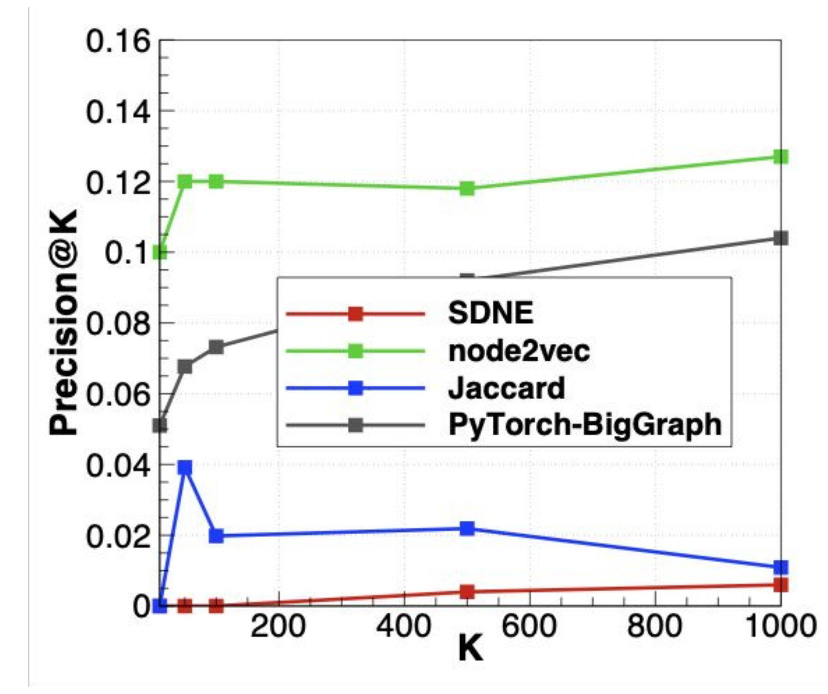
(Figures report Link Prediction precision@k)

Blog Catalog



- SDNE Parameters:
  - Embedding size: 100[1], 40[2]
  - Hidden layer size: 1000[1], 400[2]
  - Supervised loss wt: 1
  - Unsupervised loss wt: 100
- node2vec Parameters:
  - Embedding dimension: 128
  - RW length: 80
  - 10 RW per source

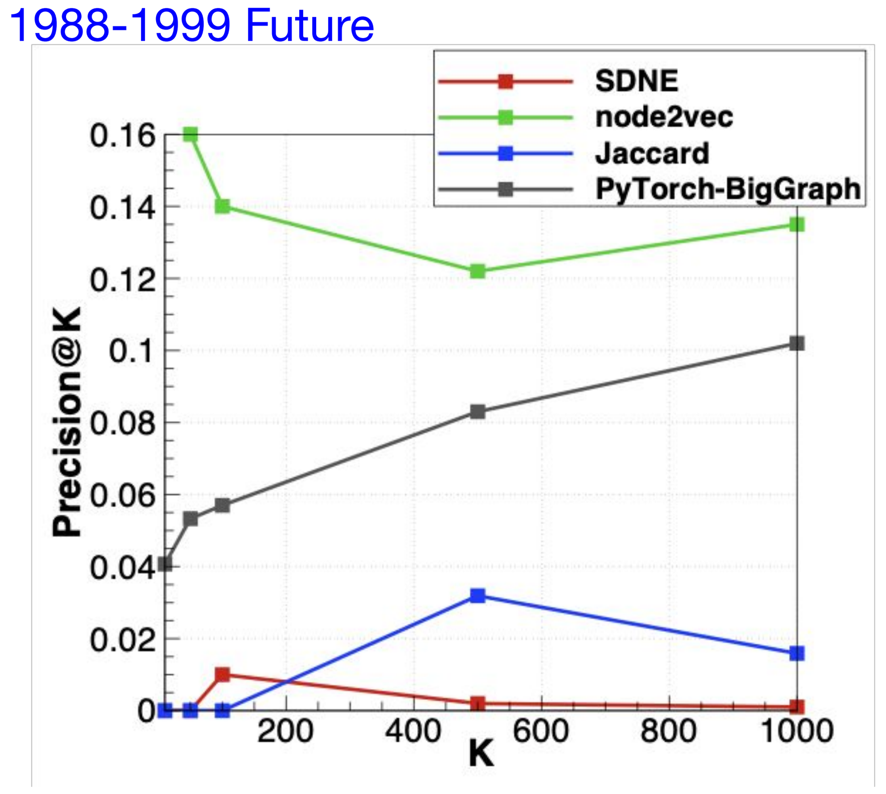
1988-1989 Baseline



SDNE vs. Node2vec:

1. node2vec: Much more computationally efficient than SDNE
2. node2vec: Performs better than SDNE, particularly on sparse graph
3. node2vec: Prediction on future is better than baseline

1988-1999 Future



Jaccard Observation:

1. Worst asymptotic complexity

BigGraph Observation:

1. Performance increases with k.
2. Highly scalable

Conclusions

- LP can work well on temporal graphs. However, sparse graphs makes LP difficult
- RW methods do disproportionately better than nn in this situation
- Scalability is challenging for all methods studied, Pytorch handles best
- ➢ Future directions: link prediction on more dense temporal graphs