On the HashGNN Node Embedding Algorithm

Jacob Sznajdman
Staff Software Engineer,
Neo4j

Adam Schill Collberg
Senior Software Engineer,
Neo4j

neo4j

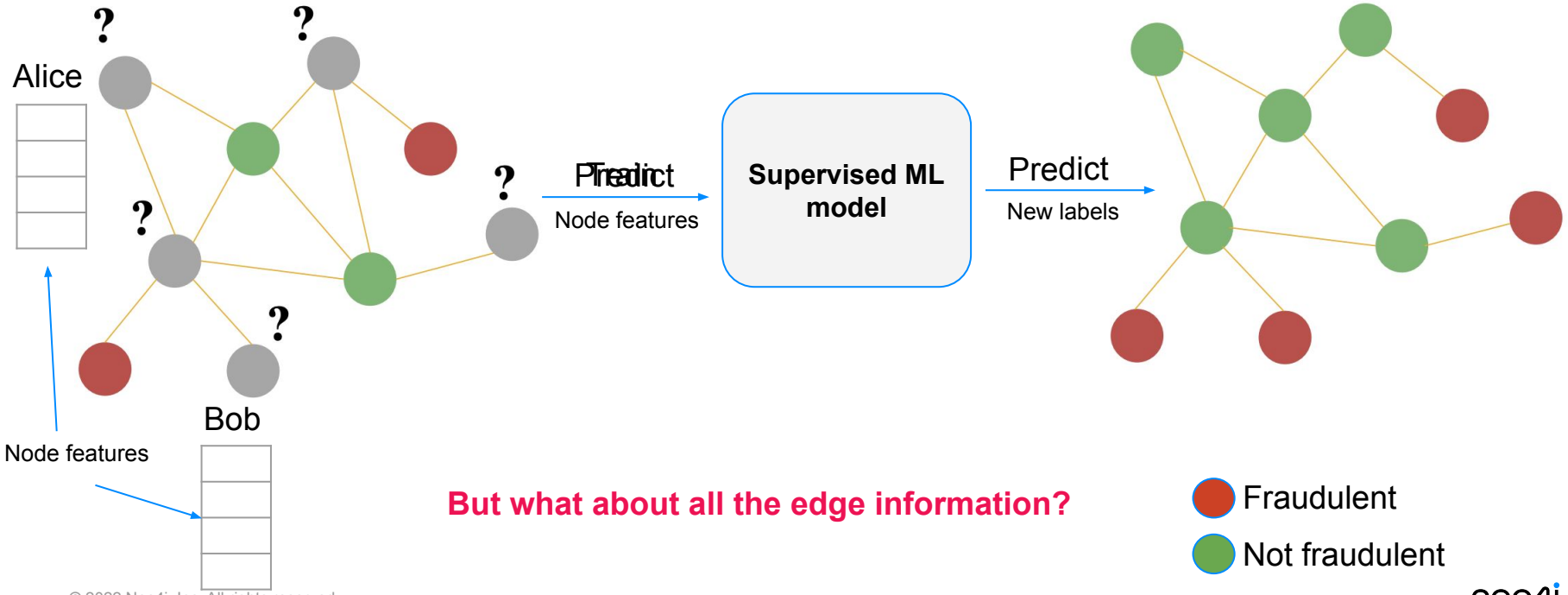
Overview

- The Node Classification Problem
- Node Embeddings
- HashGNN
  - How it works
  - Why use it?
  - Benchmarks
- Neo4j Graph Data Science
- Notebook Example
- Further Learning

The Node Classification Problem

A machine learning task on graph

Node classification



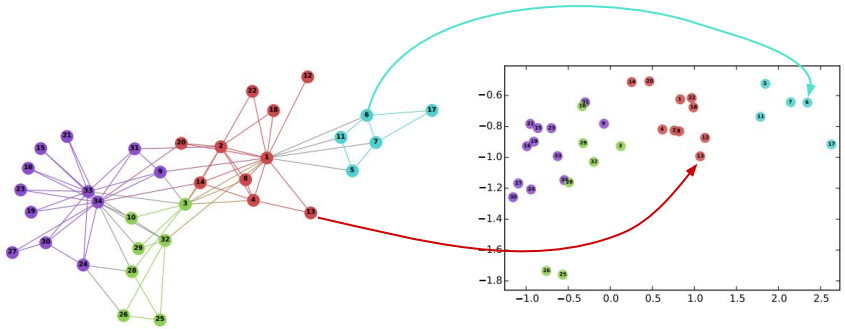
But what about all the edge information?

Node Embeddings

Encode a graph's edge information into per node vectors

Node Embeddings

- Graph topology (edges) encoded into per node vectors
- Vectors are geometric representations of nodes based on graph topology
- Suitable as input to ML models
- Analogous to word embeddings (like Word2Vec)
- Many algorithms: GNNs, Node2Vec, FastRP,...



(a) Input: Karate Graph (b) Output: Representation

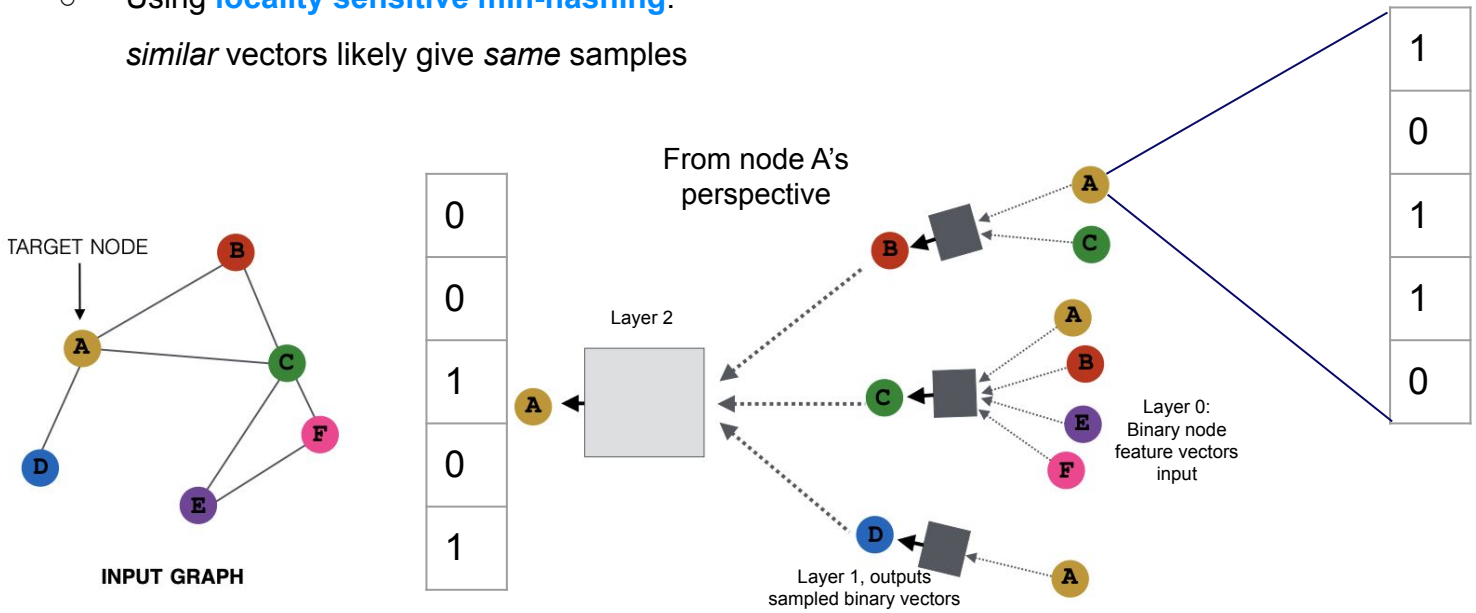
HashGNN

A node embedding algorithm from the paper:
Hashing-Accelerated Graph Neural Networks for Link Prediction,
by Wei Wu, Bin Li, Chuan Luo, Wolfgang Nejdl

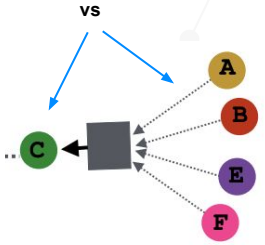
HashGNN: The original algorithm

- Takes **binary node features input**, and represents node embeddings in binary form
- Uses **message passing** along edges, similar to PageRank and many GNNs
- In the aggregation step, HashGNN **samples features** from the current node and its neighbours
  - Using **locality sensitive min-hashing**:

    similar vectors likely give *same* samples



HashGNN: Extensions



- Control **neighbors' influence**:
  - Probability of sampling features from a neighbor vs node itself

- **Heterogeneity** support:
  - Distinct hash functions for different edge types

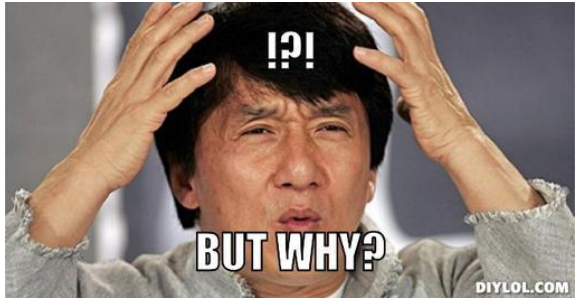
- **Binarization** of input:
  - Allow any numerical input

```
98613 = 11000000100110101
```

- Support graphs **without node features**:
  - Generate initial input internally

HashGNN: Why use it?

- Can generate **high quality** node embeddings
- Can encode the **heterogeneity** of a graph
- A lot **faster** than neural models
- **Simpler** than neural models and does not require training
- Does not require GPU
- Scales very well with increased CPU concurrency
- Does not require node feature input
- Has *inductive* capabilities
  - Though less so when using binarization in our testing so far



HashGNN: Node Classification Benchmarks

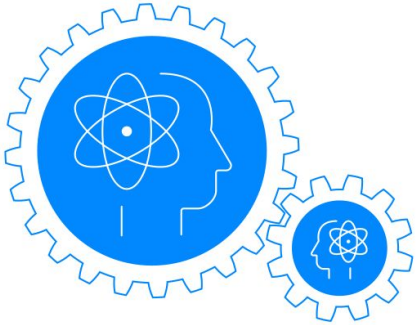
	DBLP* (% F1 score)	ACM* (% F1 score)	IMDB* (% F1 score)
DeepWalk*	63.18	67.42	32.08
metapath2vec*	85.53	87.61	35.21
GCN*	87.30	91.60	56.89
GAT*	93.71	92.33	58.14
HAN*	92.83	90.96	56.77
GTN*	**94.18**	**92.68**	**60.92**
HashGNN	93.50	92.38	59.12

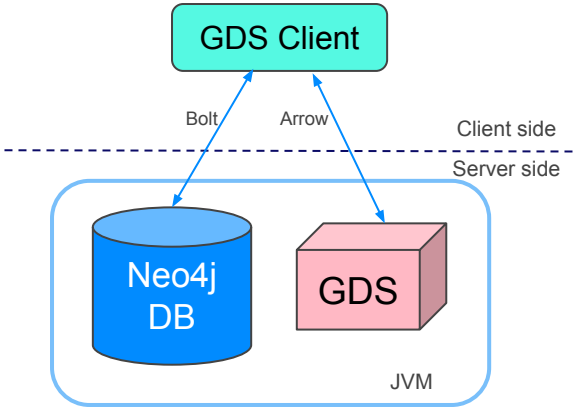
*: From the "Graph Transformer Networks" NeurIPS paper

Neo4j Graph Data Science

A Neo4j DB plugin for graph analytics at scale

Neo4j Graph Data Science (GDS) Library





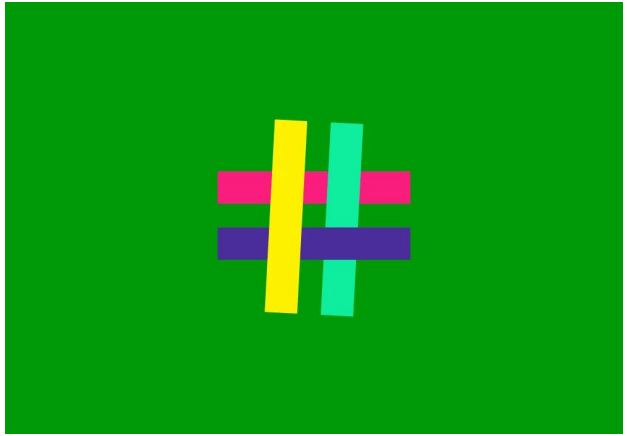
- A Neo4j DB plugin for analytics
  - GPL v3 License
  - Part of the DBMS (server) process
  - Projects graph into volatile memory for analysis
- Provides high performance graph algorithms
  - Running at scale (100s of billions of nodes)
  - **Has performant implementation of HashGNN**
- Has a Python client
  - Pythonic data science surface
  - Apache 2.0 License
  - `pip install graphdatascience`

Notebook Demo!

Heterogeneous Node Classification with HashGNN on an IMDB dataset

Further learning

- The demo notebook
- The HashGNN GDS manual docs
- The paper: Hashing-Accelerated Graph Neural Networks for Link Prediction



Thank you!

Contact us at
team-gds@neo4j.org