Graph Neural Networks: What they are and why it matters for AI applications

Kristof Neys

Department of Computer Science
Birkbeck, University of London

12 Jul 2022

Talk outline

- Why Graph Neural Networks matter
- Neural Networks - key concepts
- Graph Neural Networks
  - What are they?
  - How do they work?
  - What can they do?
- Knowledge graphs
- How to use Neo4j in your Graph journey

Why Graph Neural Networks matter...

...they matter for 300+ reasons...

Why Graph Neural Networks matter...

...they matter for 300+ reasons...we will list 3:

Why Graph Neural Networks matter...

...they matter for 300+ reasons...we will list 3:

- Reason 1: GNNs save lives!

Why Graph Neural Networks matter...

...they matter for 300+ reasons...we will list 3:

- Reason 1: GNNs save lives!
- Reason 2: GNNs solve the Traveling Salesperson Problem (sort of...)

Why Graph Neural Networks matter...

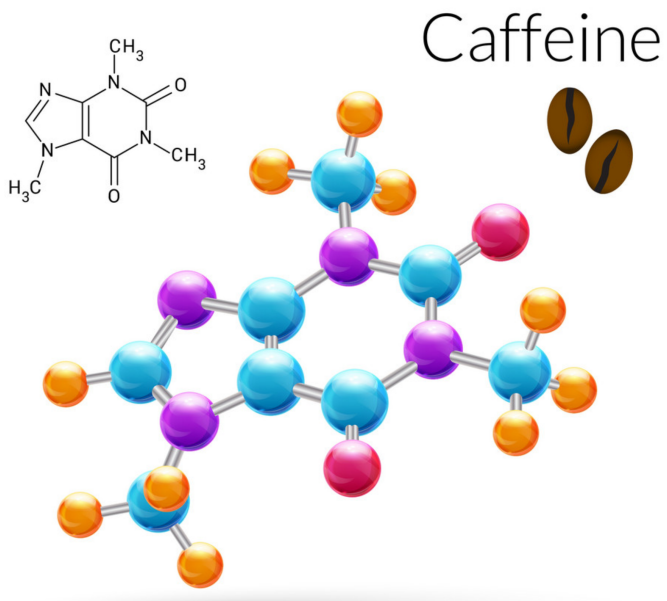
...they matter for 300+ reasons...we will list 3:

- Reason 1: GNNs save lives!
- Reason 2: GNNs solve the Traveling Salesperson Problem (sort of...)
- Reason 3: GNNs are the future of AI...

Why Graph Neural Networks matter...

Reason 1: GNNs save lives!

- Molecules are graphs...
- ....atoms as nodes, bonds as edges
- apply GNN to predict whether a molecule is a candidate for a drug



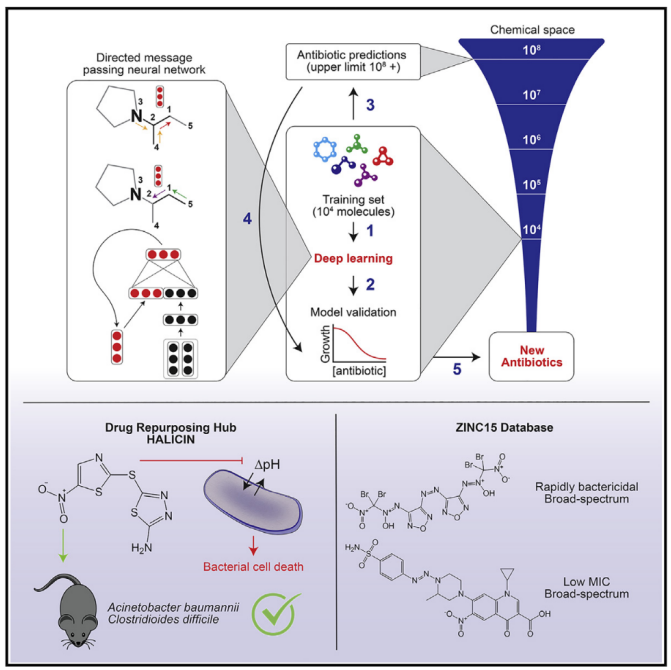
Why Graph Neural Networks matter...

Article

Cell

A Deep Learning Approach to Antibiotic Discovery

Graphical Abstract



Highlights

- A deep learning model is trained to predict antibiotics based on structure

Authors

Jonathan M. Stokes, Kevin Yang, Kyle Swanson, ..., Tommi S. Jaakkola, Regina Barzilay, James J. Collins

Correspondence

regina@csail.mit.edu (R.B.), jimjc@mit.edu (J.J.C.)

In Brief

A trained deep neural network predicts antibiotic activity in molecules that are structurally different from known antibiotics, among which Halicin exhibits efficacy against broad-spectrum bacterial infections in mice.

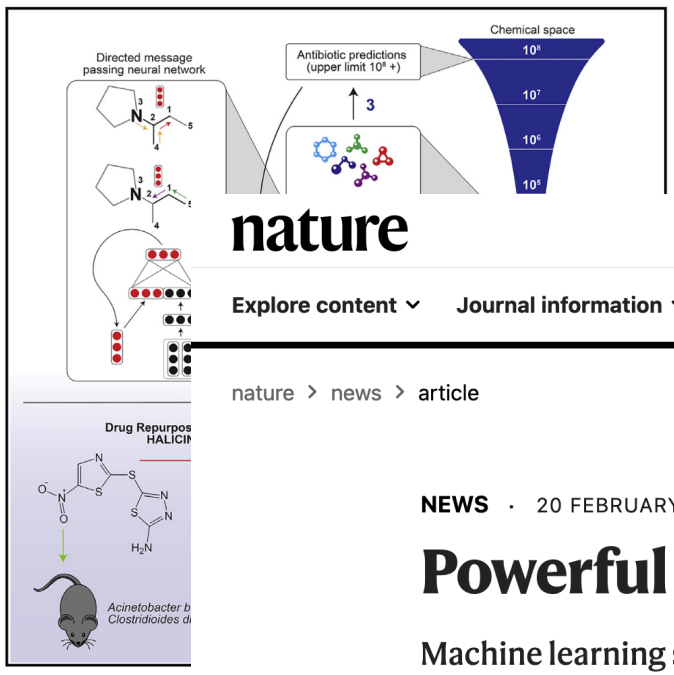
Why Graph Neural Networks matter...

Article

Cell

A Deep Learning Approach to Antibiotic Discovery

Graphical Abstract



Highlights

- A deep learning
on structure

Authors

Jonathan M. Stokes, Kevin Yang, Kyle Swanson, ..., Tommi S. Jaakkola, Regina Barzilay, James J. Collins

Correspondence

regina@csail.mit.edu (R.B.),

nature

View

Explore content · Journal information · Publish with us · Subscribe

nature › news › article

NEWS · 20 FEBRUARY 2020

Powerful antibiotics discovered using AI

Machine learning spots molecules that work even against 'untreatable' strains of bacteria.

Jo Marchant

Why Graph Neural Networks matter...

Reason 2: GNNs (almost) solve the Traveling Salesperson Problem

Why Graph Neural Networks matter...

Reason 2: GNNs (almost) solve the Traveling Salesperson Problem

- TSP problem is NP-hard, but not that hard for GNNs...
- *"...getting close to optimal results for problems up to* 100 *nodes..."*

Published as a conference paper at ICLR 2019

ATTENTION, LEARN TO SOLVE ROUTING PROBLEMS!

Wouter Kool
University of Amsterdam
ORTEC
w.w.m.kool@uva.nl

Herke van Hoof
University of Amsterdam
h.c.vanhoof@uva.nl

Max Welling
University of Amsterdam
CIFAR
m.welling@uva.nl

ABSTRACT

The recently presented idea to learn heuristics for combinatorial optimization problems is promising as it can save costly development. However, to push this idea towards practical implementation, we need better models and better ways of training. We contribute in both directions: we propose a model based on attention layers with benefits over the Pointer Network and we show how to train this model using REINFORCE with a simple baseline based on a deterministic greedy rollout, which we find is more efficient than using a value function. We significantly improve over recent learned heuristics for the Travelling Salesman Problem (TSP), getting close to optimal results for problems up to 100 nodes. With the same hyperparameters, we learn strong heuristics for two variants of the Vehicle Routing Problem (VRP), the Orienteering Problem (OP) and (a stochastic variant of) the Prize Collecting TSP (PCTSP), outperforming a wide range of baselines and getting results close to highly optimized and specialized algorithms.

Why Graph Neural Networks matter...

Reason 3: GNNs are the path to General AI...

Relational inductive biases, deep learning, and graph networks

Peter W. Battaglia[1,*], **Jessica B. Hamrick**[1], **Victor Bapst**[1], **Alvaro Sanchez-Gonzalez**[1], **Vinicius Zambaldi**[1], **Mateusz Malinowski**[1], **Andrea Tacchetti**[1], **David Raposo**[1], **Adam Santoro**[1], **Ryan Faulkner**[1], **Caglar Gulcehre**[1], **Francis Song**[1], **Andrew Ballard**[1], **Justin Gilmer**[2], **George Dahl**[2], **Ashish Vaswani**[2], **Kelsey Allen**[3], **Charles Nash**[4], **Victoria Langston**[1], **Chris Dyer**[1], **Nicolas Heess**[1], **Daan Wierstra**[1], **Pushmeet Kohli**[1], **Matt Botvinick**[1], **Oriol Vinyals**[1], **Yujia Li**[1], **Razvan Pascanu**[1]

[1]DeepMind; [2]Google Brain; [3]MIT; [4]University of Edinburgh

Abstract

Artificial intelligence (AI) has undergone a renaissance recently, making major progress in key domains such as vision, language, control, and decision-making. This has been due, in part, to cheap data and cheap compute resources, which have fit the natural strengths of deep learning. However, many defining characteristics of human intelligence, which developed under much different pressures, remain out of reach for current approaches. In particular, generalizing beyond one's experiences—a hallmark of human intelligence from infancy—remains a formidable challenge for modern AI.

The following is part position paper, part review, and part unification. We argue that combinatorial generalization must be a top priority for AI to achieve human-like abilities, and that structured representations and computations are key to realizing this objective. Just as biology uses nature and nurture cooperatively, we reject the false choice between "hand-engineering"

Why Graph Neural Networks matter...

Reason 3: GNNs are the path to General AI...

Relational inductive biases, deep learning, and graph networks

Peter W. Battaglia[1,*], **Jessica B. Hamrick**[1], **Victor Bapst**[1], **Alvaro Sanchez-Gonzalez**[1], **Vinicius Zambaldi**[1], **Mateusz Malinowski**[1], **Andrea Tacchetti**[1], **David Raposo**[1], **Adam Santoro**[1], **Ryan Faulkner**[1], **Caglar Gulcehre**[1], **Francis Song**[1], **Andrew Ballard**[1], **Justin Gilmer**[2], **George Dahl**[2], **Ashish Vaswani**[2], **Kelsey Allen**[3], **Charles Nash**[4], **Victoria Langston**[1], **Chris Dyer**[1], **Nicolas Heess**[1], **Daan Wierstra**[1], **Pushmeet Kohli**[1], **Matt Botvinick**[1], **Oriol Vinyals**[1], **Yujia Li**[1], **Razvan Pascanu**[1]

[1]DeepMind; [2]Google Brain; [3]MIT; [4]University of Edinburgh

Abstract

Artificial intelligence (AI) has undergone a renaissance recently, making major progress in key domains such as vision, language, control, and decision-making. This has been due, in part, to cheap data and cheap compute resources, which have fit the natural strengths of deep learning. However, many defining characteristics of human intelligence, which developed under much different pressures, remain out of reach for current approaches. In particular, generalizing beyond one's experiences—a hallmark of human intelligence from infancy—remains a formidable challenge for modern AI.

The following is part position paper, part review, and part unification. We argue that combinatorial generalization must be a top priority for AI to achieve human-like abilities, and that structured representations and computations are key to realizing this objective. Just as biology uses nature and nurture cooperatively, we reject the false choice between "hand-engineering"

- *"We argue that combinatorial generalization must be a top priority for AI to achieve human-like abilities, and that structured representations and computations are key to realizing this objective."*

Why Graph Neural Networks matter...

Reason 3: GNNs are the path to General AI...

Relational inductive biases, deep learning, and graph networks

Peter W. Battaglia[1]*, Jessica B. Hamrick[1], Victor Bapst[1],
Alvaro Sanchez-Gonzalez[1], Vinicius Zambaldi[1], Mateusz Malinowski[1],
Andrea Tacchetti[1], David Raposo[1], Adam Santoro[1], Ryan Faulkner[1],
Caglar Gulcehre[1], Francis Song[1], Andrew Ballard[1], Justin Gilmer[2],

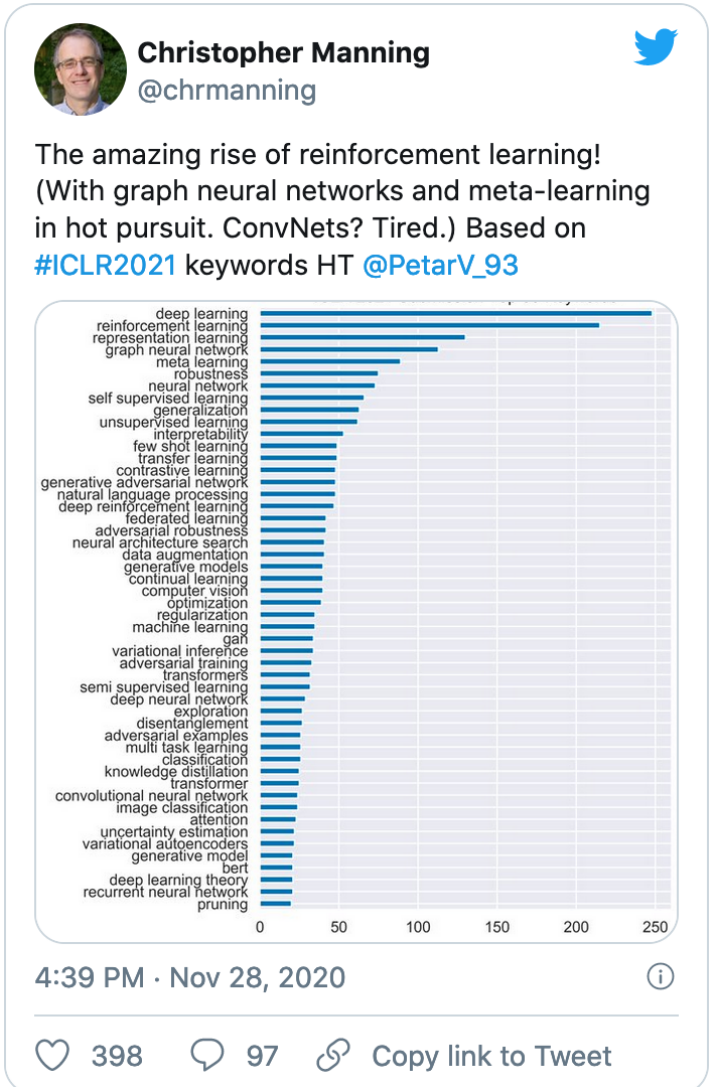
Intro to DeepMind's Graph-Nets

A short overview of the core components of Graph-Nets

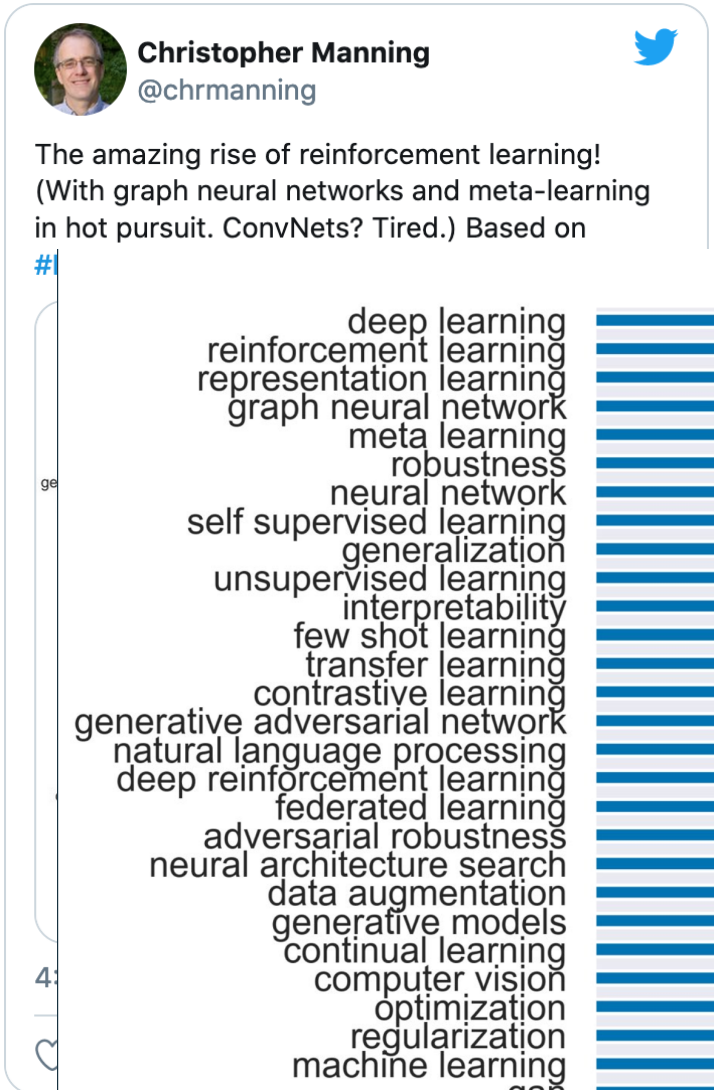
Kristof Neys Jan 20 · 7 min read

n must be a top
es, and that
ns are key to

No surprise then: GNNs are HOT...



No surprise then: GNNs are HOT...



Christopher Manning
@chrmanning

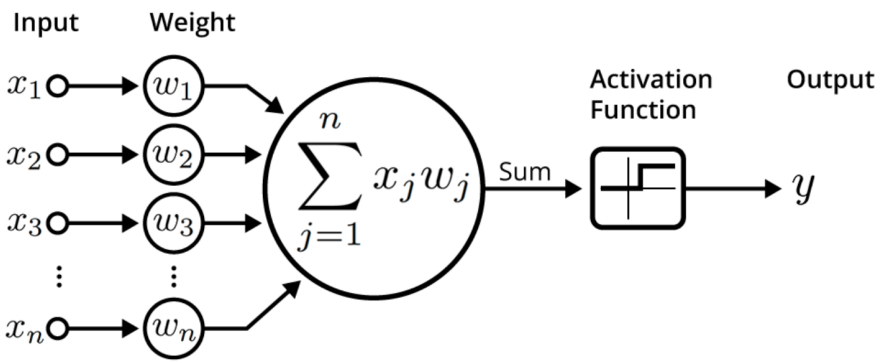
The amazing rise of reinforcement learning! (With graph neural networks and meta-learning in hot pursuit. ConvNets? Tired.) Based on

Review of Neural Networks

Neural Networks

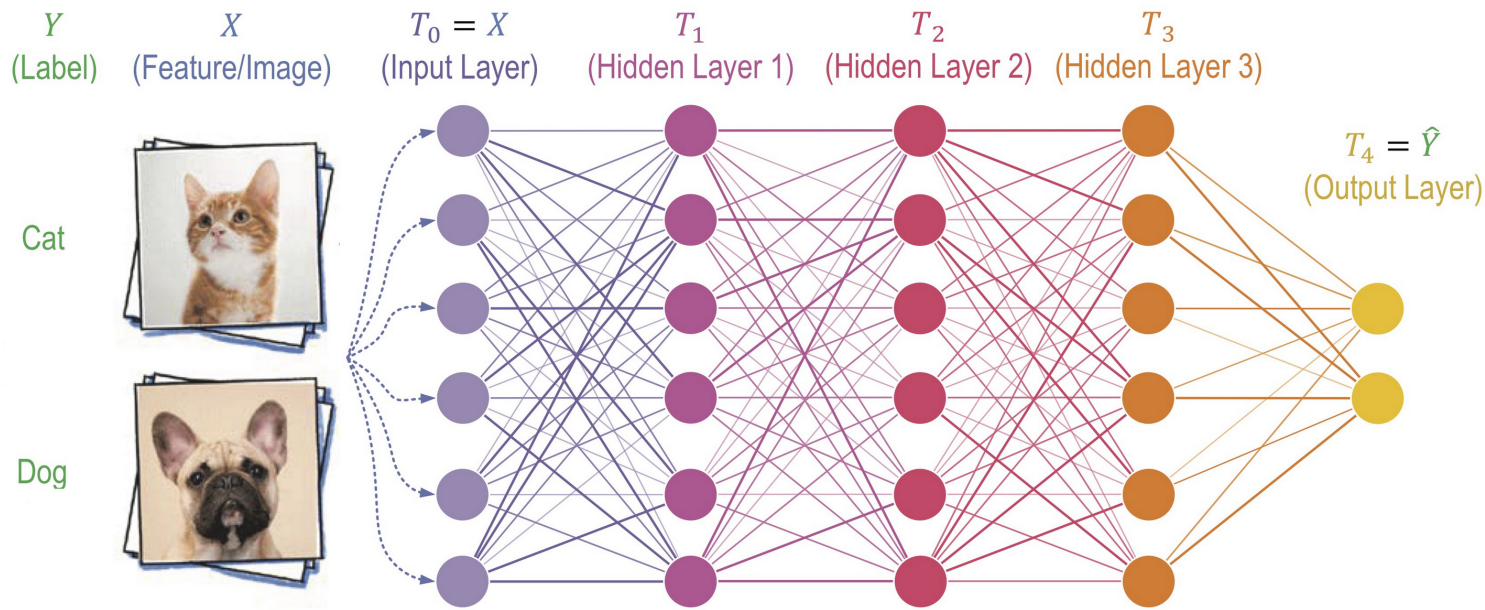
- Basic intuition is one of a succession of increasingly refined **data filters**
- Two key components: artificial neuron and layer
- The artificial neuron is an information processing unit taking in a set of inputs which are multiplied by numerical weights
- A series of these neurons can be stacked in a single layer
- A Multilayer Perceptron (MLP) combines several layers to perform an output prediction

An Artificial Neuron



- Receive a list of numbers as input, x_i
- Multiply each number in the list by its weight, w_i
- Add up all of the results to get a single number
- Apply a non-linear *activation function*
- The weights, w_i, are the learnable parameters

Neural Networks - Cat or Dog?



- Training data will only give a label for the final output layer (i.e. Cat, Dog). The intermediate layers are called *hidden layers*
- The neural network learns through measuring the output against a *loss function* and updating the weight parameters using the *backpropagation algorithm*

Graph Neural Networks

Graph Neural Networks

- What are they?

Graph Neural Networks

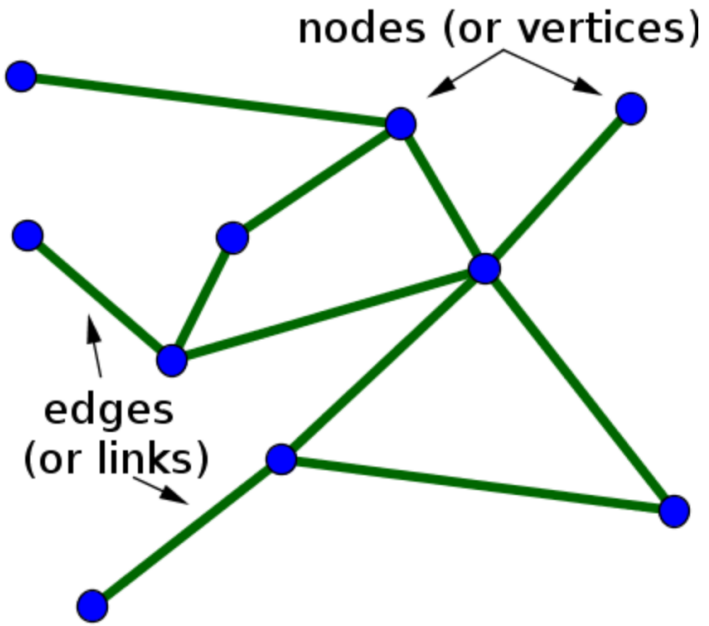
- What are they?
- How do they work?

Graph Neural Networks

- What are they?
- How do they work?
- What can they do?

Graphs

- $G = (V, E)$
- V being the set of nodes, and E the set of edges



Graph Neural Networks - what are they?

- The core idea is to generate representations of nodes that actually depend on the structure of the graph, as well as any feature information we might have

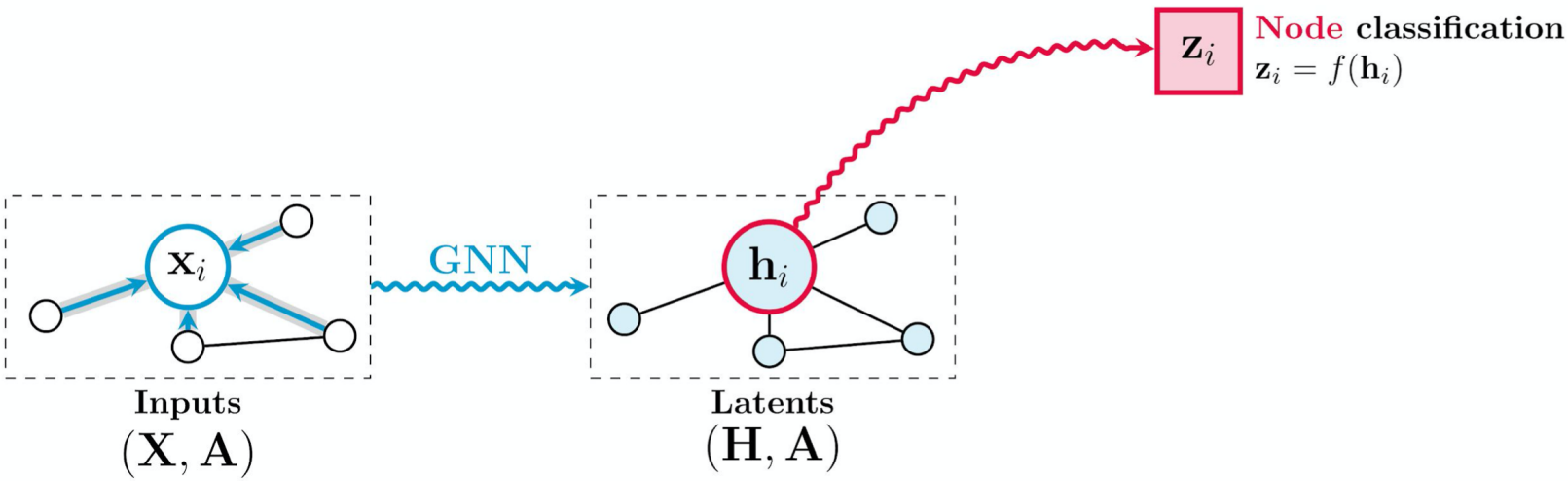
Graph Neural Networks - what are they?

- The core idea is to generate representations of nodes that actually depend on the structure of the graph, as well as any feature information we might have
- The key challenge is that the usual deep learning methods as illustrated on the previous slides only work on Euclidean data, i.e grid-like or sequential data structures

Graph Neural Networks - what are they?

- The core idea is to generate representations of nodes that actually depend on the structure of the graph, as well as any feature information we might have
- The key challenge is that the usual deep learning methods as illustrated on the previous slides only work on Euclidean data, i.e grid-like or sequential data structures
- The essence of GNNs is the message passing framework upon which most GNNs are based

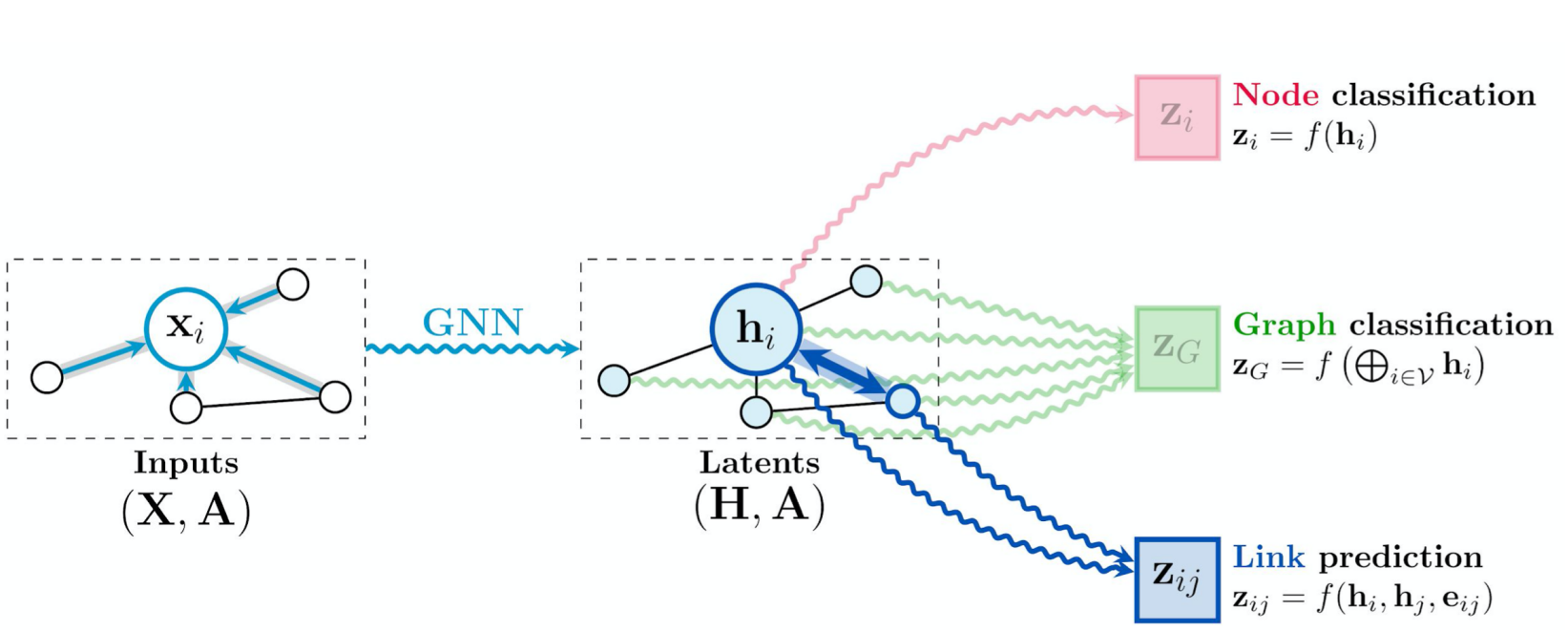
Graph Neural Networks - what can they do?



[1]

[1] image by Dr. Petar Veličković

Graph Neural Networks - what can they do?



[2]

[2]image by Dr. Petar Veličković

Graph Neural Networks - definition & workings

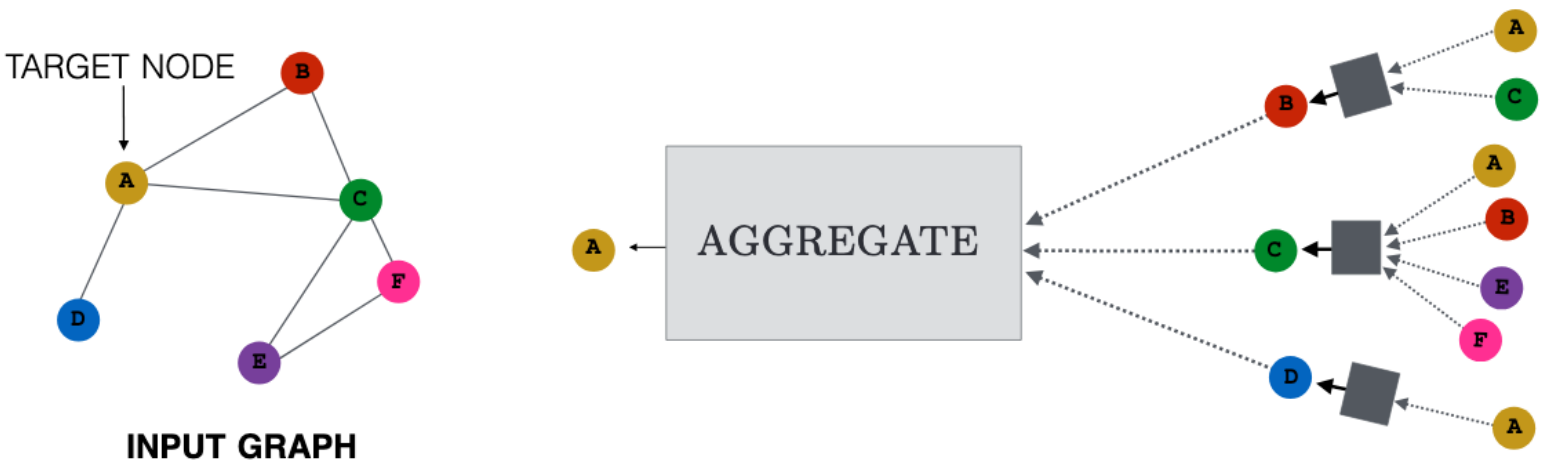
Definition (Graph Neural Network)

Graph neural networks (GNNs) are a class of NN models suitable for processing graph structured data

- The architecture of a GNN is structured according to a graph $G = (V, E)$
- A GNN takes as input an instance of a graph, G, where the nodes are associated with feature vectors, $\mathbf{x}_i$ and edges can also be associated with feature vectors; $\mathbf{x}_{ij}$
- Hidden representations of the nodes and edges in the neural network are denoted by $\mathbf{h}_i$ and $\mathbf{h}_{ij}$, respectively.
- A *message passing* update is executed in sequence to obtain updated node and edge representations $\mathbf{h}'_i$ and $\mathbf{h}'_{ij}$, respectively.

GNN - How do they work?

- Core idea: during each message passing iteration in a GNN a hidden embedding $\mathbf{h}_a^{(k)}$, $a \in V$, is **updated** according to information **aggregated** from $a's$ graph neighborhood $\mathcal{N}(a)$.



GNN - Message Passing framework

- The basic intuition is that at each iteration, (k), every node aggregates information from its local neighborhood

GNN - Message Passing framework

- The basic intuition is that at each iteration, (k), every node aggregates information from its local neighborhood
- The key functions are the `UPDATE` and `AGGREGATE` functions. It's the `UPDATE` function that provides trainable parameter matrices, equivalent to the MLP framework

GNN - Message Passing framework

- The basic intuition is that at each iteration, (k), every node aggregates information from its local neighborhood
- The key functions are the `UPDATE` and `AGGREGATE` functions. It's the `UPDATE` function that provides trainable parameter matrices, equivalent to the MLP framework
- In essence, we are learning a mapping between a feature vector, which is an embedding of the neighbourhood of a node, and the label associated with that node.

GNN - Message Passing framework

- Formally, this is represented as follows:

$$\mathbf{h}_a^{(k+1)} = \texttt{UPDATE}^k \left(\mathbf{h}_a^{(k)}, \texttt{AGGREGATE}^k \left(\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(a) \right) \right)$$

where UPDATE and AGGREGATE are differentiable functions.

GNN - Message Passing framework

- Formally, this is represented as follows:

$$\mathbf{h}_a^{(k+1)} = \texttt{UPDATE}^k \left(\mathbf{h}_a^{(k)}, \texttt{AGGREGATE}^k \left(\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(a) \right) \right)$$

  where UPDATE and AGGREGATE are differentiable functions.

- Let $\mathbf{m}_{\mathcal{N}(a)}$ be the *"message"* that is aggregated from $a's$ neighborhood $\mathcal{N}(a)$, then:

$$\mathbf{m}_{\mathcal{N}(a)} = \texttt{AGGREGATE}^k \left(\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(a) \right)$$

GNN - Message Passing framework

- The `UPDATE` function combines $\mathbf{m}_{\mathcal{N}_{(a)}}$ with the previous embedding $\mathbf{h}_a^k$ of node a and computes $\mathbf{h}_a^{k+1}$
- `UPDATE`, in the basic GNN is defined as:

$$\sigma\left(\mathbf{W}_a\mathbf{h}_a + \mathbf{W}_{neigh}\mathbf{m}_{\mathcal{N}_{(a)}}\right)$$

- $\mathbf{W}_a$ and $\mathbf{W}_{neigh}$ form the trainable weight matrices
- The prediction gets measured in a loss function and a backpropagation algorithm is applied to update the weight matrices

Aggregate & Update define the GNN

Name	Variant	Aggregator	Updater
Spectral Methods	ChebNet	$\mathbf{N}_k = \mathbf{T}_k(\tilde{\mathbf{L}})\mathbf{X}$	$\mathbf{H} = \sum_{k=0}^{K} \mathbf{N}_k \mathbf{\Theta}_k$
	1^{st}-order model	$\mathbf{N}_0 = \mathbf{X}$ $\mathbf{N}_1 = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\mathbf{X}$	$\mathbf{H} = \mathbf{N}_0\mathbf{\Theta}_0 + \mathbf{N}_1\mathbf{\Theta}_1$
	Single parameter	$\mathbf{N} = (\mathbf{I}_N + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{X}$	$\mathbf{H} = \mathbf{N}\mathbf{\Theta}$
	GCN	$\mathbf{N} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}$	$\mathbf{H} = \mathbf{N}\mathbf{\Theta}$
Non-spectral Methods	Neural FPs	$\mathbf{h}^t_{\mathcal{N}_v} = \mathbf{h}^{t-1}_v + \sum_{k=1}^{\mathcal{N}_v} \mathbf{h}^{t-1}_k$	$\mathbf{h}^t_v = \sigma(\mathbf{h}^t_{\mathcal{N}_v}\mathbf{W}^{\mathcal{N}_v}_L)$
	DCNN	Node classification: $\mathbf{N} = \mathbf{P}^*\mathbf{X}$ Graph classification: $\mathbf{N} = 1^T_N\mathbf{P}^*\mathbf{X}/N$	$\mathbf{H} = f\left(\mathbf{W}^c \odot \mathbf{N}\right)$
	GraphSAGE	$\mathbf{h}^t_{\mathcal{N}_v} = \text{AGGREGATE}_t\left(\{\mathbf{h}^{t-1}_u, \forall u \in \mathcal{N}_v\}\right)$	$\mathbf{h}^t_v = \sigma\left(\mathbf{W}^t \cdot [\mathbf{h}^{t-1}_v \| \mathbf{h}^t_{\mathcal{N}_v}]\right)$
Graph Attention Networks	GAT	$\alpha_{vk} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_k]\right)\right)}{\sum_{j\in\mathcal{N}_v}\exp(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_v \| \mathbf{W}\mathbf{h}_j]))}$ $\mathbf{h}^t_{\mathcal{N}_v} = \sigma\left(\sum_{k\in\mathcal{N}_v}\alpha_{vk}\mathbf{W}\mathbf{h}_k\right)$ Multi-head concatenation: $\mathbf{h}^t_{\mathcal{N}_v} = \Big\|_{m=1}^{M}\sigma\left(\sum_{k\in\mathcal{N}_v}\alpha^m_{vk}\mathbf{W}^m\mathbf{h}_k\right)$ Multi-head average: $\mathbf{h}^t_{\mathcal{N}_v} = \sigma\left(\frac{1}{M}\sum_{m=1}^{M}\sum_{k\in\mathcal{N}_v}\alpha^m_{vk}\mathbf{W}^m\mathbf{h}_k\right)$	$\mathbf{h}^t_v = \mathbf{h}^t_{\mathcal{N}_v}$
Gated Graph Neural Networks	GGNN	$\mathbf{h}^t_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{h}^{t-1}_k + \mathbf{b}$	$\mathbf{z}^t_v = \sigma(\mathbf{W}^z\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{U}^z\mathbf{h}^{t-1}_v)$ $\mathbf{r}^t_v = \sigma(\mathbf{W}^r\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{U}^r\mathbf{h}^{t-1}_v)$ $\widetilde{\mathbf{h}^t_v} = \tanh(\mathbf{W}\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{U}(\mathbf{r}^t_v \odot \mathbf{h}^{t-1}_v))$ $\mathbf{h}^t_v = (1 - \mathbf{z}^t_v) \odot \mathbf{h}^{t-1}_v + \mathbf{z}^t_v \odot \widetilde{\mathbf{h}^t_v}$
Graph LSTM	Tree LSTM (Child sum)	$\mathbf{h}^t_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{h}^{t-1}_k$	$\mathbf{i}^t_v = \sigma(\mathbf{W}^i\mathbf{x}^t_v + \mathbf{U}^i\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{b}^i)$ $\mathbf{f}^t_{vk} = \sigma\left(\mathbf{W}^f\mathbf{x}^t_v + \mathbf{U}^f\mathbf{h}^{t-1}_k + \mathbf{b}^f\right)$ $\mathbf{o}^t_v = \sigma(\mathbf{W}^o\mathbf{x}^t_v + \mathbf{U}^o\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{b}^o)$ $\mathbf{u}^t_v = \tanh(\mathbf{W}^u\mathbf{x}^t_v + \mathbf{U}^u\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{b}^u)$ $\mathbf{c}^t_v = \mathbf{i}^t_v \odot \mathbf{u}^t_v + \sum_{k\in\mathcal{N}_v}\mathbf{f}^t_{vk} \odot \mathbf{c}^{t-1}_k$ $\mathbf{h}^t_v = \mathbf{o}^t_v \odot \tanh(\mathbf{c}^t_v)$
	Tree LSTM (N-ary)	$\mathbf{h}^{ti}_{\mathcal{N}_v} = \sum_{l=1}^{K}\mathbf{U}^i_l\mathbf{h}^{t-1}_{vl}$ $\mathbf{h}^{tf}_{\mathcal{N}_v k} = \sum_{l=1}^{K}\mathbf{U}^f_{kl}\mathbf{h}^{t-1}_{vl}$ $\mathbf{h}^{to}_{\mathcal{N}_v} = \sum_{l=1}^{K}\mathbf{U}^o_l\mathbf{h}^{t-1}_{vl}$ $\mathbf{h}^{tu}_{\mathcal{N}_v} = \sum_{l=1}^{K}\mathbf{U}^u_l\mathbf{h}^{t-1}_{vl}$	$\mathbf{i}^t_v = \sigma(\mathbf{W}^i\mathbf{x}^t_v + \mathbf{h}^{ti}_{\mathcal{N}_v} + \mathbf{b}^i)$ $\mathbf{f}^t_{vk} = \sigma(\mathbf{W}^f\mathbf{x}^t_v + \mathbf{h}^{tf}_{\mathcal{N}_v k} + \mathbf{b}^f)$ $\mathbf{o}^t_v = \sigma(\mathbf{W}^o\mathbf{x}^t_v + \mathbf{h}^{to}_{\mathcal{N}_v} + \mathbf{b}^o)$ $\mathbf{u}^t_v = \tanh(\mathbf{W}^u\mathbf{x}^t_v + \mathbf{h}^{tu}_{\mathcal{N}_v} + \mathbf{b}^u)$ $\mathbf{c}^t_v = \mathbf{i}^t_v \odot \mathbf{u}^t_v + \sum_{l=1}^{K}\mathbf{f}^t_{vl} \odot \mathbf{c}^{t-1}_{vl}$ $\mathbf{h}^t_v = \mathbf{o}^t_v \odot \tanh(\mathbf{c}^t_v)$
	Graph LSTM in [44]	$\mathbf{h}^{ti}_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{U}^i_{m(v,k)}\mathbf{h}^{t-1}_k$ $\mathbf{h}^{to}_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{U}^o_{m(v,k)}\mathbf{h}^{t-1}_k$ $\mathbf{h}^{tu}_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{U}^u_{m(v,k)}\mathbf{h}^{t-1}_k$	$\mathbf{i}^t_v = \sigma(\mathbf{W}^i\mathbf{x}^t_v + \mathbf{h}^{ti}_{\mathcal{N}_v} + \mathbf{b}^i)$ $\mathbf{f}^t_{vk} = \sigma(\mathbf{W}^f\mathbf{x}^t_v + \mathbf{U}^f_{m(v,k)}\mathbf{h}^{t-1}_k + \mathbf{b}^f)$ $\mathbf{o}^t_v = \sigma(\mathbf{W}^o\mathbf{x}^t_v + \mathbf{h}^{to}_{\mathcal{N}_v} + \mathbf{b}^o)$ $\mathbf{u}^t_v = \tanh(\mathbf{W}^u\mathbf{x}^t_v + \mathbf{h}^{tu}_{\mathcal{N}_v} + \mathbf{b}^u)$ $\mathbf{c}^t_v = \mathbf{i}^t_v \odot \mathbf{u}^t_v + \sum_{k\in\mathcal{N}_v}\mathbf{f}^t_{vk} \odot \mathbf{c}^{t-1}_k$ $\mathbf{h}^t_v = \mathbf{o}^t_v \odot \tanh(\mathbf{c}^t_v)$

Message Passing Framework - summary

The message passing framework is analogous to a standard MLP as it relies on linear operations followed by an activation function

Message Passing Framework - summary

The message passing framework is analogous to a standard MLP as it relies on linear operations followed by an activation function

- we first sum the messages incoming from the neighbours

Message Passing Framework - summary

The message passing framework is analogous to a standard MLP as it relies on linear operations followed by an activation function

- we first sum the messages incoming from the neighbours
- we combine the neighborhood information with the node's previous embedding using a linear combination

Message Passing Framework - summary

The message passing framework is analogous to a standard MLP as it relies on linear operations followed by an activation function

- we first sum the messages incoming from the neighbours
- we combine the neighborhood information with the node's previous embedding using a linear combination
- apply an elementwise nonlinearity, i.e. an activation function

GNNs - What can they do?

Three broad use cases:

GNNs - What can they do?

Three broad use cases:

- node classification

GNNs - What can they do?

Three broad use cases:

- node classification
- link (i.e. edge) prediction

GNNs - What can they do?

Three broad use cases:

- node classification
- link (i.e. edge) prediction
- graph classification

Example: Graph Attention Networks

- Cora dataset; the MNIST of graph land
  - 2708 academic papers, 7 classes ("Theory", "Reinforcement learning", "Probabilistic methods",...) 5429 edges = citations, 1433 feature vector per node extracted from the text
  - only give 140 labeled nodes to train on (20 per class)
  - use GNN to predict the classes on 1000 nodes
- Semi-supervised learning to perform node classification

GNNs - performance on Cora dataset

Method	Cora
MLP	55.1%
ManiReg (Belkin et al., 2006)	59.5%
SemiEmb (Weston et al., 2012)	59.0%
LP (Zhu et al., 2003)	68.0%
DeepWalk (Perozzi et al., 2014)	67.2%
ICA (Lu & Getoor, 2003)	75.1%
Planetoid (Yang et al., 2016)	75.7%
Chebyshev (Defferrard et al., 2016)	81.2%
GCN (Kipf & Welling, 2017)	81.5%
MoNet (Monti et al., 2016)	$81.7 \pm 0.5\%$
GCN-64*	$81.4 \pm 0.5\%$
GAT (ours)	$\mathbf{83.0} \pm 0.7\%$

GNNs - performance on Cora dataset

Method	Cora
MLP	55.1%
ManiReg (Belkin et al., 2006)	59.5%
SemiEmb (Weston et al., 2012)	59.0%
LP (Zhu et al., 2003)	68.0%
DeepWalk (Perozzi et al., 2014)	67.2%
ICA (Lu & Getoor, 2003)	75.1%
Planetoid (Yang et al., 2016)	75.7%
Chebyshev (Defferrard et al., 2016)	81.2%
GCN (Kipf & Welling, 2017)	81.5%
MoNet (Monti et al., 2016)	$81.7 \pm 0.5\%$
GCN-64*	$81.4 \pm 0.5\%$
GAT (ours)	$\mathbf{83.0} \pm 0.7\%$

- State-of-the-Art GNNs offer 28% improvement over MLPs...

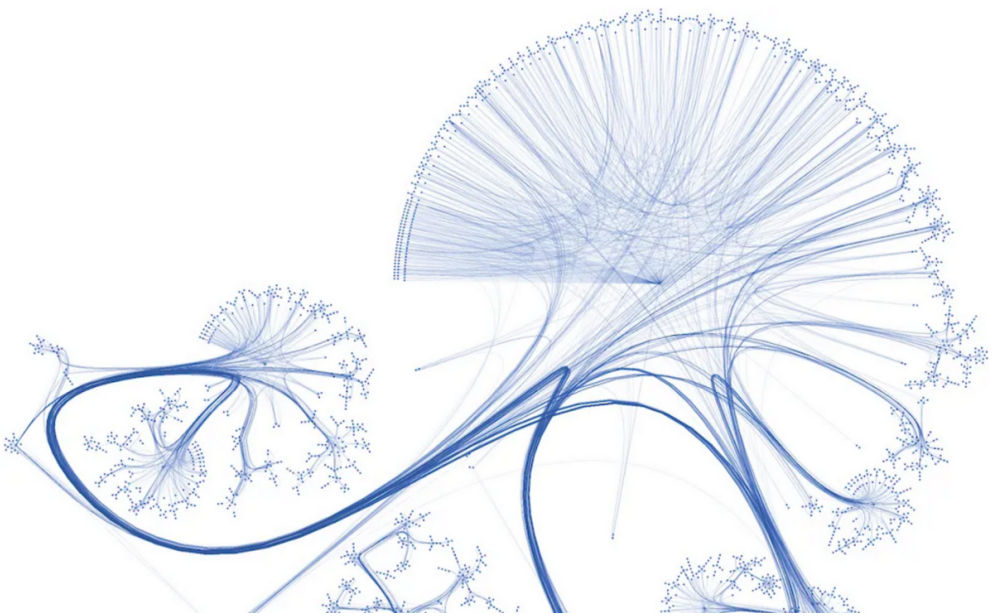
Graph Attention Network

Neo4j & DGL — a seamless integration

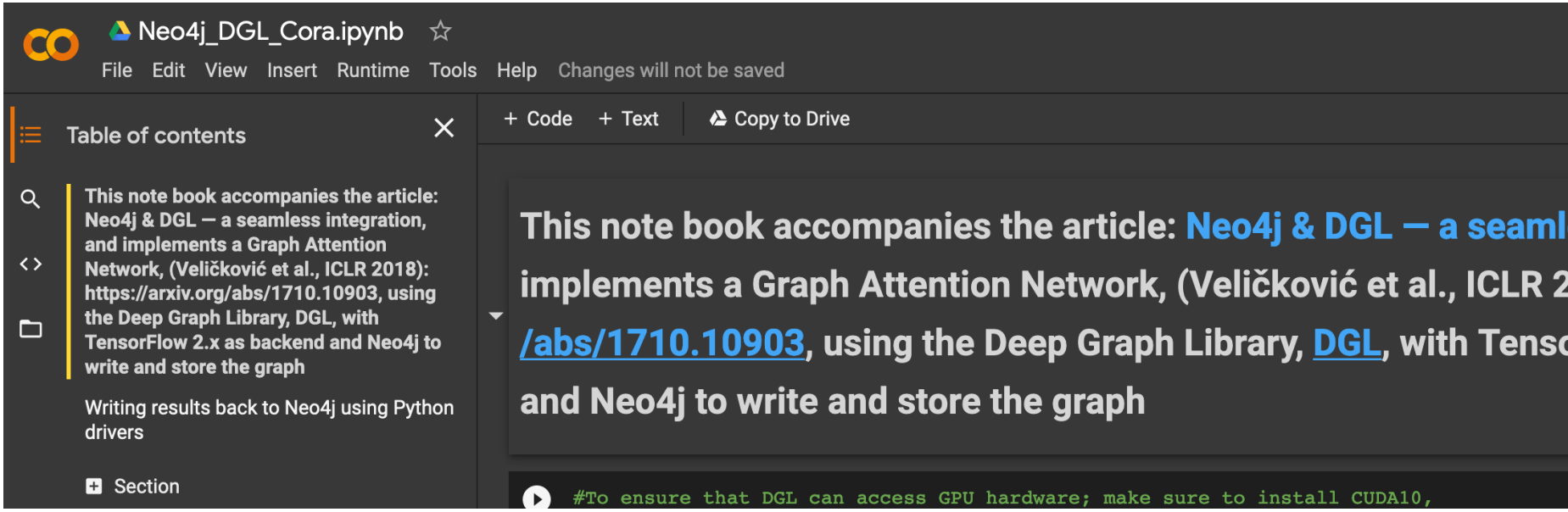
In this article we will illustrate how to integrate a Graph Attention Network model using the Deep Graph Library into the Neo4j workflow and deploying the Neo4j Python driver.

Kristof Neys Mar 5 · 11 min read

This blog post was co-authored with Clair Sullivan and Mark Needham



Graph Attention Network



Knowledge Graphs

Knowledge Graphs

Christopher Strachey in letter to Alan Turing

"I am convinced that the crux of the problem of learning is recognizing relationships and being able to use them"

Knowledge graphs

Definition (Knowledge graphs)

A knowledge graph is a structured representation of facts, consisting of entities, relationships and semantic descriptions

- Entities can be real-world objects and abstract concepts

Knowledge graphs

Definition (Knowledge graphs)

A knowledge graph is a structured representation of facts, consisting of entities, relationships and semantic descriptions

- Entities can be real-world objects and abstract concepts
- Relationships represent the relation between entities

Knowledge graphs

Definition (Knowledge graphs)

A knowledge graph is a structured representation of facts, consisting of entities, relationships and semantic descriptions

- Entities can be real-world objects and abstract concepts
- Relationships represent the relation between entities
- Semantic description of the entities and relationships contain types and properties with a well-defined meaning

Knowledge graphs

Definition (Knowledge graphs)

A knowledge graph is a structured representation of facts, consisting of entities, relationships and semantic descriptions

- Entities can be real-world objects and abstract concepts
- Relationships represent the relation between entities
- Semantic description of the entities and relationships contain types and properties with a well-defined meaning
- A knowledge graph is a graphical representation of a knowledge base

Knowledge graphs

Definition (Knowledge graphs)

A knowledge graph is a structured representation of facts, consisting of entities, relationships and semantic descriptions

- Entities can be real-world objects and abstract concepts
- Relationships represent the relation between entities
- Semantic description of the entities and relationships contain types and properties with a well-defined meaning
- A knowledge graph is a graphical representation of a knowledge base
- A knowledge base is in the form of Resource Description Framework, i.e. (Elon_Musk, makes, cars), or Property Graph

KG + ML = a marriage made in heaven...

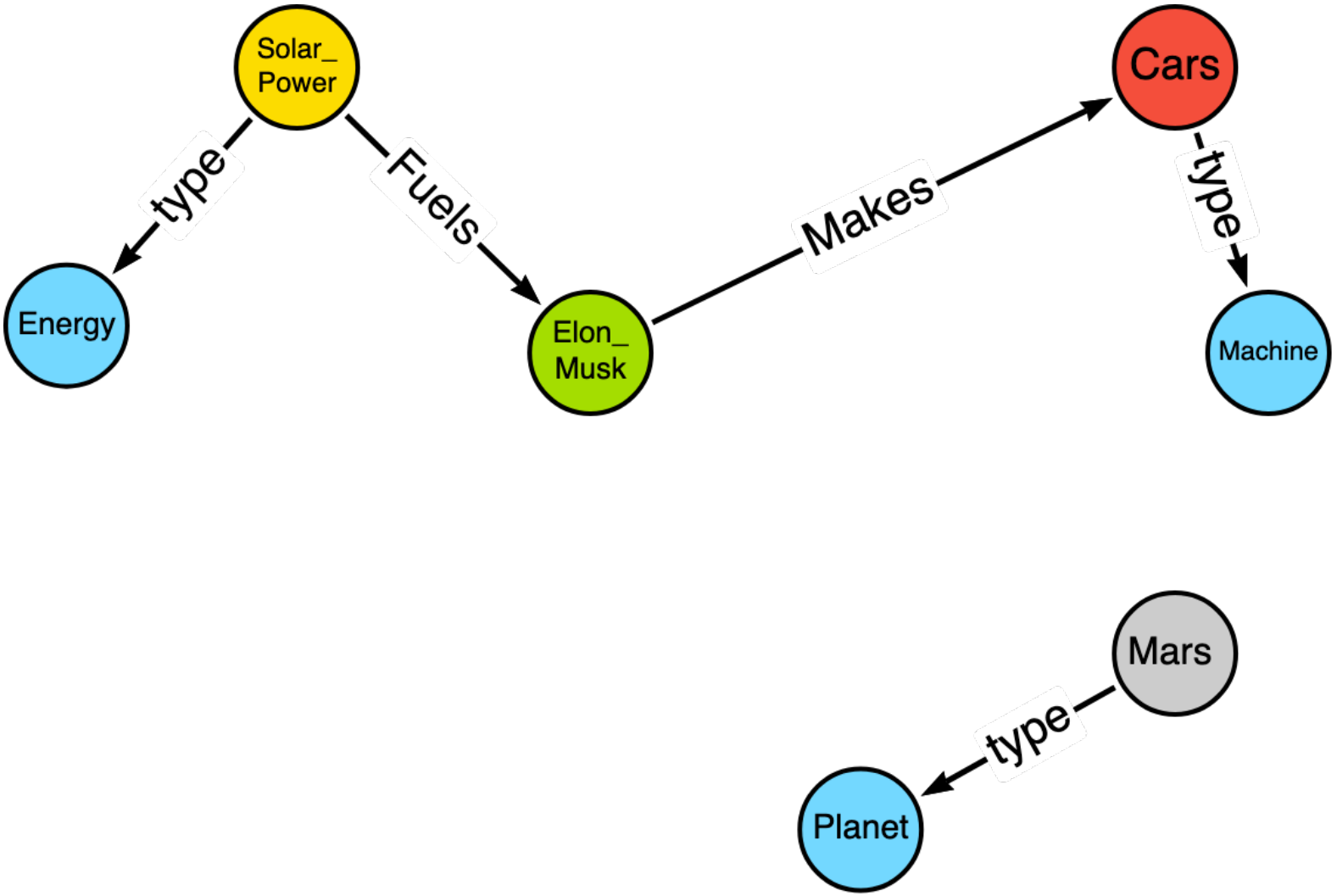
- Developing machine learning models is important since even the largest knowledge bases are incomplete and as such the gap in knowledge limit the downstream applications

KG + ML = a marriage made in heaven...

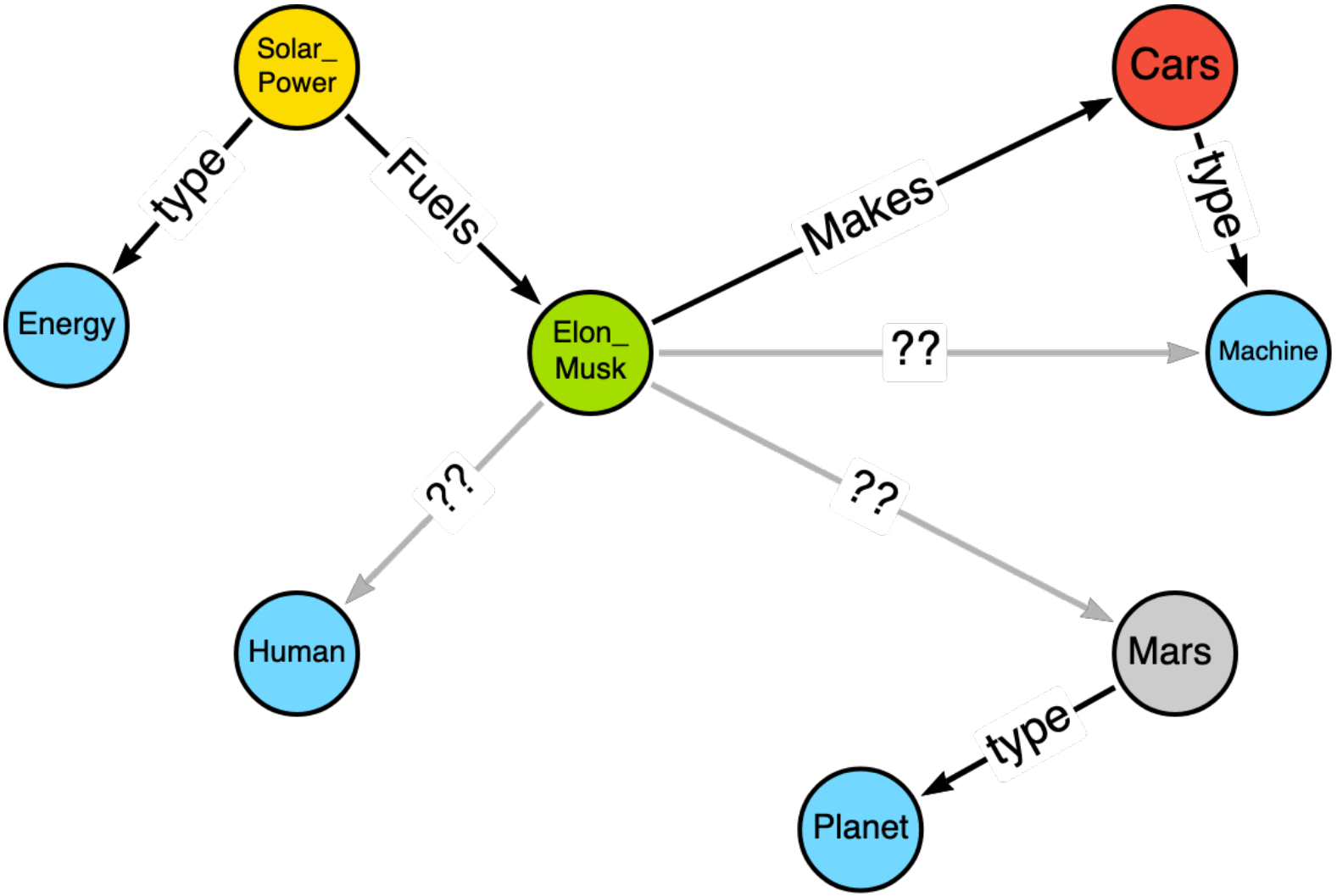
- Developing machine learning models is important since even the largest knowledge bases are incomplete and as such the gap in knowledge limit the downstream applications

- Predicting missing information in knowledge bases using machine learning is "Statistical Relational Learning", a subfield of ML that deals with learning with relational structure

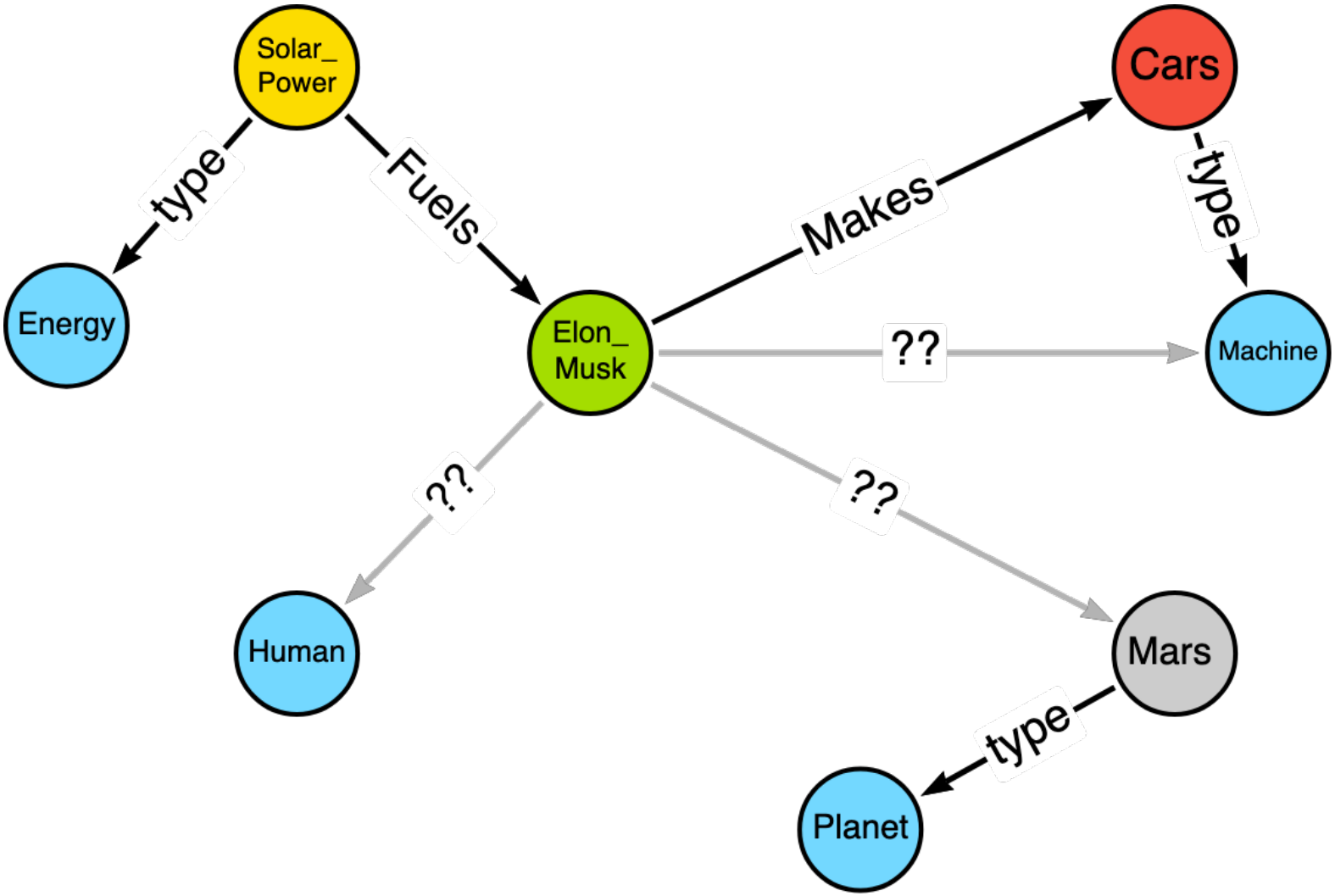
Simplifying...



Knowledge graph completion

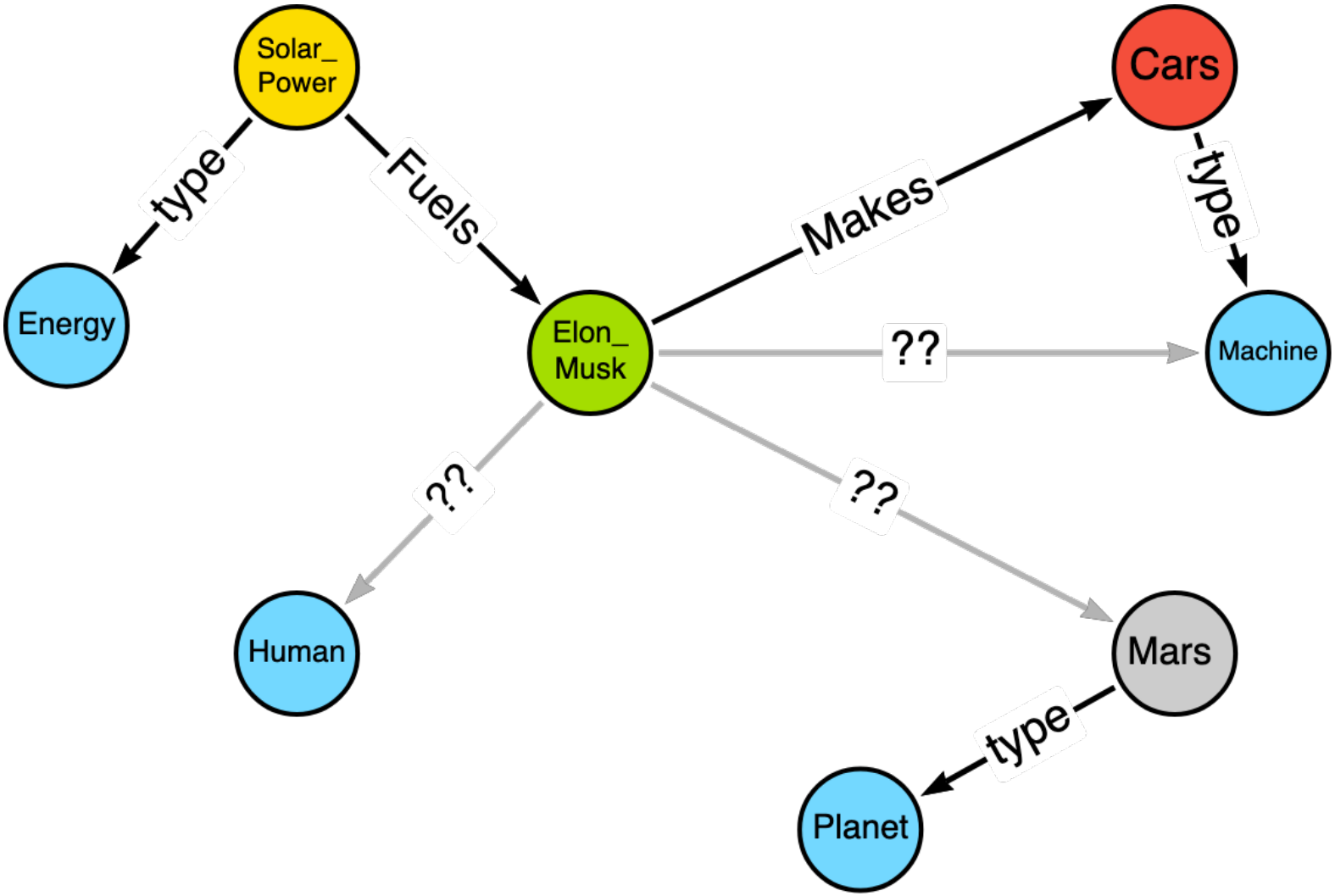


Knowledge graph completion



- Elon Musk.... Human or Machine?

Knowledge graph completion



- Elon Musk.... Human or Machine?
- ...born on Mars, Holiday...?

Knowledge graphs - history

- popularized by Google in 2012...

Knowledge graphs - history

- popularized by Google in 2012...
- 500 billion facts (i.e. triples)....

Knowledge graphs - history

- popularized by Google in 2012...
- 500 billion facts (i.e. triples)....
- on 5 billion entities...

Knowledge graphs - history

- popularized by Google in 2012...
- 500 billion facts (i.e. triples)....
- on 5 billion entities...
- applications: Q&A (i.e. Alexa), search, recommender systems, ∞...

Knowledge graphs - history

- popularized by Google in 2012...
- 500 billion facts (i.e. triples)....
- on 5 billion entities...
- applications: Q&A (i.e. Alexa), search, recommender systems, ∞...

Alan Turing

Mathematician

Alan Mathison Turing OBE FRS was an English mathematician, computer scientist, logician, cryptanalyst, philosopher, and theoretical biologist. Wikipedia

Born: 23 June 1912, Maida Vale, London

Died: 7 June 1954, Wilmslow

Education: Princeton University (1936–1938), MORE

Known for: Cryptanalysis of the Enigma, Turing's proof, MORE

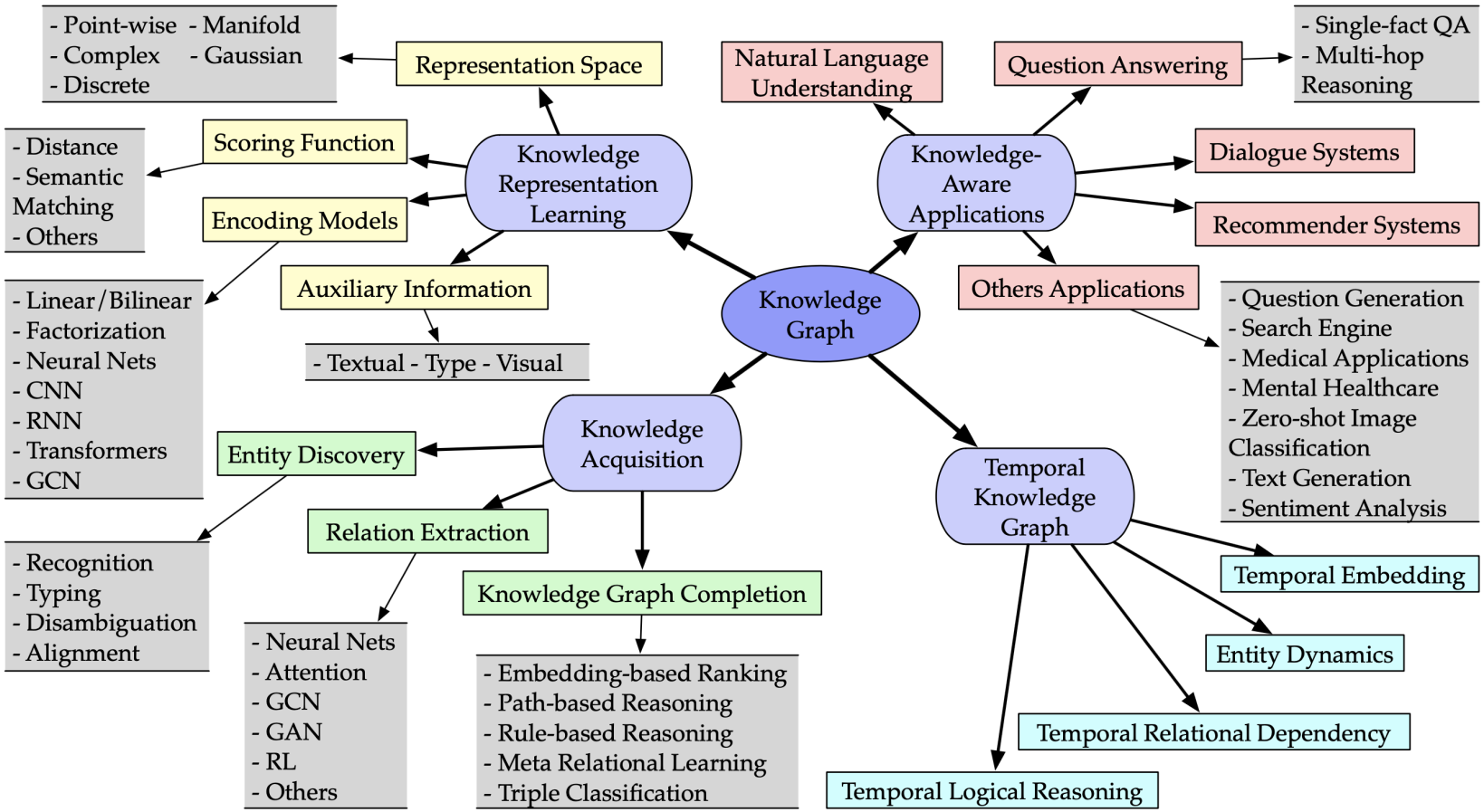
Quotes View 4+ more

We can only see a short distance ahead, but we can see plenty there that needs to be done.

I propose to consider the question, 'Can machines think?

Science is a differential equation. Religion is a boundary condition.

Driving explosive growth in research...



...resulting in high accuracy

Method	Filtered						Extra features
	WN18			FB15k			
	MR	H10	MRR	MR	H10	MRR	
SE (Bordes et al., 2011)	985	80.5	-	162	39.8	-	None
Unstructured (Bordes et al., 2014)	304	38.2	-	979	6.3	-	
TransE (Bordes et al., 2013)	251	89.2	-	125	47.1	-	
TransH (Wang et al., 2014)	303	86.7	-	87	64.4	-	
TransR (Lin et al., 2015b)	225	92.0	-	77	68.7	-	
CTransR (Lin et al., 2015b)	218	92.3	-	75	70.2	-	
KG2E (He et al., 2015)	331	92.8	-	59	74.0	-	
TransD (Ji et al., 2015)	212	92.2	-	91	77.3	-	
lppTransD (Yoon et al., 2016)	270	94.3	-	78	78.7	-	
TranSparse (Ji et al., 2016)	211	93.2	-	82	79.5	-	
TATEC (Garcia-Duran et al., 2016)	-	-	-	58	76.7	-	
NTN (Socher et al., 2013)	-	66.1	0.53	-	41.4	0.25	
HolE (Nickel et al., 2016)	-	94.9	**0.938**	-	73.9	0.524	
STransE (Nguyen et al., 2016)	**206**	93.4	0.657	69	79.7	0.543	
ComplEx (Trouillon et al., 2017)	-	94.7	**0.941**	-	84.0	0.692	
ProjE wlistwise (Shi and Weniger, 2017)	-	-	-	**34**	88.4	-	
IRN (Shen et al., 2016)	249	**95.3**	-	**38**	**92.7**	-	
RTransE (García-Durán et al., 2015)	-	-	-	50	76.2	-	Path
PTransE (Lin et al., 2015a)	-	-	-	58	84.6	-	
GAKE (Feng et al., 2015)	-	-	-	119	64.8	-	
Gaifman (Niepert, 2016)	352	93.9	-	75	84.2	-	
Hiri (Liu et al., 2016)	-	90.8	0.691	-	70.3	0.603	
R-GCN+ (Schlichtkrull et al., 2017)	-	**96.4**	0.819	-	84.2	0.696	
NLFeat (Toutanova and Chen, 2015)	-	94.3	**0.940**	-	87.0	**0.822**	Text
TEKE_H (Wang and Li, 2016)	**114**	92.9	-	108	73.0	-	
SSP (Xiao et al., 2017)	**156**	93.2	-	82	79.0	-	
DistMult (orig) (Yang et al., 2015)	-	94.2	0.83	-	57.7	0.35	None
DistMult (Toutanova and Chen, 2015)	-	-	-	-	79.7	0.555	
DistMult (Trouillon et al., 2017)	-	93.6	0.822	-	82.4	0.654	
Single DistMult (this work)	655	94.6	0.797	42.2	**89.3**	**0.798**	
Ensemble DistMult (this work)	457	**95.0**	0.790	**35.9**	**90.4**	**0.837**	

Important papers to highlight

Graph Neural Networks strike again...

Modeling Relational Data with Graph Convolutional Networks

Michael Schlichtkrull*
University of Amsterdam
m.s.schlichtkrull@uva.nl

Thomas N. Kipf*
University of Amsterdam
t.n.kipf@uva.nl

Peter Bloem
VU Amsterdam
p.bloem@vu.nl

Rianne van den Berg
University of Amsterdam
r.vandenberg@uva.nl

Ivan Titov
University of Amsterdam
titov@uva.nl

Max Welling
University of Amsterdam, CIFAR†
m.welling@uva.nl

Abstract

Knowledge graphs enable a wide variety of applications, including question answering and information retrieval. Despite the great effort invested in their creation and maintenance, even the largest (e.g., Yago, DBPedia or Wikidata) remain incomplete. We introduce Relational Graph Convolutional Networks (R-GCNs) and apply them to two standard knowledge base completion tasks: Link prediction (recovery of missing facts, i.e. subject-predicate-object triples) and entity classification (recovery of missing entity attributes). R-GCNs are related to a recent class of neural networks operating on graphs, and are developed specifically to deal with the highly multi-relational data characteristic of realistic knowledge bases. We demonstrate the effectiveness of R-GCNs as a stand-alone model for entity classification. We further show that factorization models for link prediction such as DistMult can be significantly improved by enriching them with an encoder model to accumulate evidence over multiple inference

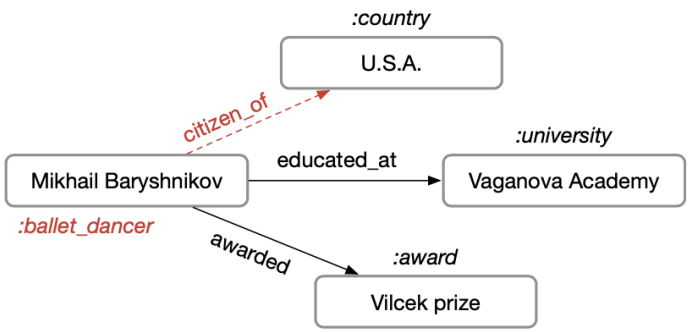


Figure 1: A knowledge base fragment: The nodes are entities, the edges are relations labeled with their types, the nodes are labeled with entity types (e.g., *university*). The edge and the node label shown in red are the missing information to be inferred.

Relational Graph Convolutional Networks (R-GCNs)

- Recall: GCN's compute node embeddings..

Modeling Relational Data with Graph Convolutional Networks

Michael Schlichtkrull[*]
University of Amsterdam
m.s.schlichtkrull@uva.nl

Thomas N. Kipf[*]
University of Amsterdam
t.n.kipf@uva.nl

Peter Bloem
VU Amsterdam
p.bloem@vu.nl

Rianne van den Berg
University of Amsterdam
r.vandenberg@uva.nl

Ivan Titov
University of Amsterdam
titov@uva.nl

Max Welling
University of Amsterdam, CIFAR[†]
m.welling@uva.nl

Abstract

Knowledge graphs enable a wide variety of applications, including question answering and information retrieval. Despite the great effort invested in their creation and maintenance, even the largest (e.g., Yago, DBPedia or Wikidata) remain incomplete. We introduce Relational Graph Convolutional Networks (R-GCNs) and apply them to two standard knowledge base completion tasks: Link prediction (recovery of missing facts, i.e. subject-predicate-object triples) and entity classification (recovery of missing entity attributes). R-GCNs are related to a recent class of neural networks operating on graphs, and are developed specifically to deal with the highly multi-relational data characteristic of realistic knowledge bases. We demonstrate the effectiveness of R-GCNs as a stand-alone model for entity classification. We further show that factorization models for link prediction such as DistMult can be significantly improved by enriching them with an encoder model to accumulate evidence over multiple inference

Figure 1: A knowledge base fragment: The nodes are entities, the edges are relations labeled with their types, the nodes are labeled with entity types (e.g., *university*). The edge and the node label shown in red are the missing information to be inferred.

Relational Graph Convolutional Networks (R-GCNs)

Modeling Relational Data with Graph Convolutional Networks

Michael Schlichtkrull[*]
University of Amsterdam
m.s.schlichtkrull@uva.nl

Thomas N. Kipf[*]
University of Amsterdam
t.n.kipf@uva.nl

Peter Bloem
VU Amsterdam
p.bloem@vu.nl

Rianne van den Berg
University of Amsterdam
r.vandenberg@uva.nl

Ivan Titov
University of Amsterdam
titov@uva.nl

Max Welling
University of Amsterdam, CIFAR[†]
m.welling@uva.nl

Abstract

Knowledge graphs enable a wide variety of applications, including question answering and information retrieval. Despite the great effort invested in their creation and maintenance, even the largest (e.g., Yago, DBPedia or Wikidata) remain incomplete. We introduce Relational Graph Convolutional Networks (R-GCNs) and apply them to two standard knowledge base completion tasks: Link prediction (recovery of missing facts, i.e. subject-predicate-object triples) and entity classification (recovery of missing entity attributes). R-GCNs are related to a recent class of neural networks operating on graphs, and are developed specifically to deal with the highly multi-relational data characteristic of realistic knowledge bases. We demonstrate the effectiveness of R-GCNs as a stand-alone model for entity classification. We further show that factorization models for link prediction such as DistMult can be significantly improved by enriching them with an encoder model to accumulate evidence over multiple inference

Figure 1: A knowledge base fragment: The nodes are entities, the edges are relations labeled with their types, the nodes are labeled with entity types (e.g., *university*). The edge and the node label shown in red are the missing information to be inferred.

- Recall: GCN's compute node embeddings..
- extend GCN to multi-relational graphs (i.e. Knowledge graphs)

Relational Graph Convolutional Networks (R-GCNs)

Modeling Relational Data with Graph Convolutional Networks

Michael Schlichtkrull[*] University of Amsterdam m.s.schlichtkrull@uva.nl

Thomas N. Kipf[*] University of Amsterdam t.n.kipf@uva.nl

Peter Bloem VU Amsterdam p.bloem@vu.nl

Rianne van den Berg University of Amsterdam r.vandenberg@uva.nl

Ivan Titov University of Amsterdam titov@uva.nl

Max Welling University of Amsterdam, CIFAR[†] m.welling@uva.nl

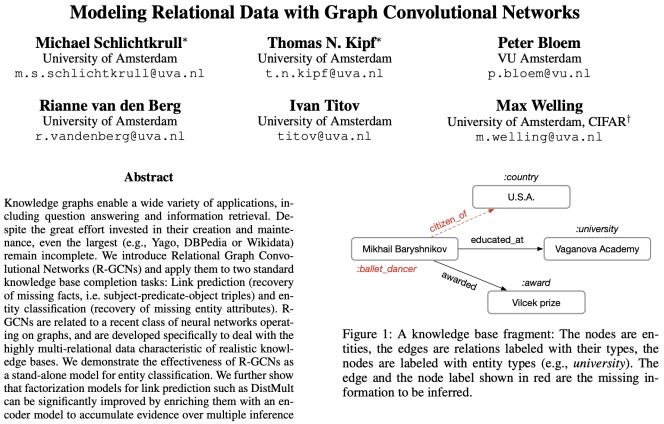
Abstract

Knowledge graphs enable a wide variety of applications, including question answering and information retrieval. Despite the great effort invested in their creation and maintenance, even the largest (e.g., Yago, DBPedia or Wikidata) remain incomplete. We introduce Relational Graph Convolutional Networks (R-GCNs) and apply them to two standard knowledge base completion tasks: Link prediction (recovery of missing facts, i.e. subject-predicate-object triples) and entity classification (recovery of missing entity attributes). R-GCNs are related to a recent class of neural networks operating on graphs, and are developed specifically to deal with the highly multi-relational data characteristic of realistic knowledge bases. We demonstrate the effectiveness of R-GCNs as a stand-alone model for entity classification. We further show that factorization models for link prediction such as DistMult can be significantly improved by enriching them with an encoder model to accumulate evidence over multiple inference

Figure 1: A knowledge base fragment: The nodes are entities, the edges are relations labeled with their types, the nodes are labeled with entity types (e.g., *university*). The edge and the node label shown in red are the missing information to be inferred.

- Recall: GCN's compute node embeddings..
- extend GCN to multi-relational graphs (i.e. Knowledge graphs)
- to perform link prediction and entity classification

Relational Graph Convolutional Networks (R-GCNs)



Modeling Relational Data with Graph Convolutional Networks

Michael Schlichtkrull[*] University of Amsterdam m.s.schlichtkrull@uva.nl

Thomas N. Kipf[*] University of Amsterdam t.n.kipf@uva.nl

Peter Bloem VU Amsterdam p.bloem@vu.nl

Rianne van den Berg University of Amsterdam r.vandenberg@uva.nl

Ivan Titov University of Amsterdam titov@uva.nl

Max Welling University of Amsterdam, CIFAR[†] m.welling@uva.nl

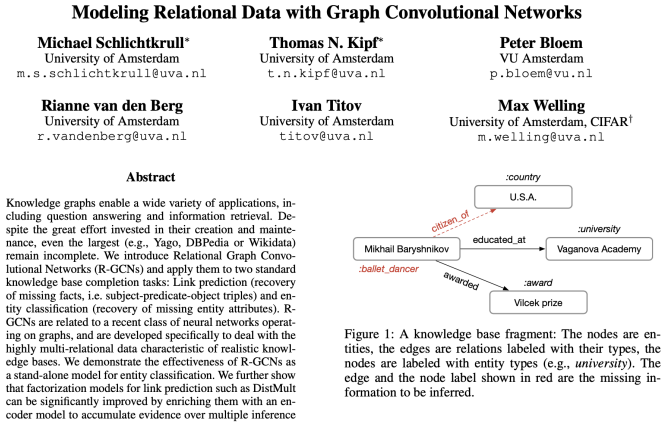
Abstract

Knowledge graphs enable a wide variety of applications, including question answering and information retrieval. Despite the great effort invested in their creation and maintenance, even the largest (e.g., Yago, DBPedia or Wikidata) remain incomplete. We introduce Relational Graph Convolutional Networks (R-GCNs) and apply them to two standard knowledge base completion tasks: Link prediction (recovery of missing facts, i.e. subject-predicate-object triples) and entity classification (recovery of missing entity attributes). R-GCNs are related to a recent class of neural networks operating on graphs, and are developed specifically to deal with the highly multi-relational data characteristic of realistic knowledge bases. We demonstrate the effectiveness of R-GCNs as a stand-alone model for entity classification. We further show that factorization models for link prediction such as DistMult can be significantly improved by enriching them with an encoder model to accumulate evidence over multiple inference

Figure 1: A knowledge base fragment: The nodes are entities, the edges are relations labeled with their types, the nodes are labeled with entity types (e.g., *university*). The edge and the node label shown in red are the missing information to be inferred.

- Recall: GCN's compute node embeddings..
- extend GCN to multi-relational graphs (i.e. Knowledge graphs)
- to perform link prediction and entity classification
- extend existing factorization models with the embedding method

Relational Graph Convolutional Networks (R-GCNs)



Modeling Relational Data with Graph Convolutional Networks

Michael Schlichtkrull[*] University of Amsterdam m.s.schlichtkrull@uva.nl

Thomas N. Kipf[*] University of Amsterdam t.n.kipf@uva.nl

Peter Bloem VU Amsterdam p.bloem@vu.nl

Rianne van den Berg University of Amsterdam r.vandenberg@uva.nl

Ivan Titov University of Amsterdam titov@uva.nl

Max Welling University of Amsterdam, CIFAR[†] m.welling@uva.nl

Abstract

Knowledge graphs enable a wide variety of applications, including question answering and information retrieval. Despite the great effort invested in their creation and maintenance, even the largest (e.g., Yago, DBPedia or Wikidata) remain incomplete. We introduce Relational Graph Convolutional Networks (R-GCNs) and apply them to two standard knowledge base completion tasks: Link prediction (recovery of missing facts, i.e. subject-predicate-object triples) and entity classification (recovery of missing entity attributes). R-GCNs are related to a recent class of neural networks operating on graphs, and are developed specifically to deal with the highly multi-relational data characteristic of realistic knowledge bases. We demonstrate the effectiveness of R-GCNs as a stand-alone model for entity classification. We further show that factorization models for link prediction such as DistMult can be significantly improved by enriching them with an encoder model to accumulate evidence over multiple inference

Figure 1: A knowledge base fragment: The nodes are entities, the edges are relations labeled with their types, the nodes are labeled with entity types (e.g., *university*). The edge and the node label shown in red are the missing information to be inferred.

- Recall: GCN's compute node embeddings..
- extend GCN to multi-relational graphs (i.e. Knowledge graphs)
- to perform link prediction and entity classification
- extend existing factorization models with the embedding method
- ...and improve link prediction by 30%!

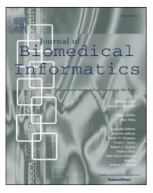
Drug discovery with Knowledge graphs

Journal of Biomedical Informatics 115 (2021) 103696

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

journal homepage: www.elsevier.com/locate/yjbin



Original Research

Drug repurposing for COVID-19 via knowledge graph completion

Rui Zhang [a,*], Dimitar Hristovski [b,1], Dalton Schutte [a,1], Andrej Kastrin [b,1], Marcelo Fiszman [c], Halil Kilicoglu [d]

[a] Institute for Health Informatics and Department of Pharmaceutical Care & Health Systems, University of Minnesota, MN, USA
[b] Institute for Biostatistics and Medical Informatics, Faculty of Medicine, University of Ljubljana, Ljubljana, Slovenia
[c] NITES - Núcleo de Inovação e Tecnologia Em Saúde, Pontifical Catholic University of Rio de Janeiro, Brazil
[d] School of Information Sciences, University of Illinois at Urbana-Champaign, Champaign, IL, USA

ARTICLE INFO

Keywords:
COVID-19
Drug repurposing
Knowledge graph completion
Literature-based discovery
Text mining

ABSTRACT

Objective: To discover candidate drugs to repurpose for COVID-19 using literature-derived knowledge and knowledge graph completion methods.
Methods: We propose a novel, integrative, and neural network-based literature-based discovery (LBD) approach to identify drug candidates from PubMed and other COVID-19-focused research literature. Our approach relies on semantic triples extracted using SemRep (via SemMedDB). We identified an informative and accurate subset of semantic triples using filtering rules and an accuracy classifier developed on a BERT variant. We used this subset to construct a knowledge graph, and applied five state-of-the-art, neural knowledge graph completion algorithms (i.e., TransE, RotatE, DistMult, ComplEx, and STELP) to predict drug repurposing candidates. The models were trained and assessed using a time slicing approach and the predicted drugs were compared with a list of drugs reported in the literature and evaluated in clinical trials. These models were complemented by a discovery pattern-based approach.
Results: Accuracy classifier based on PubMedBERT achieved the best performance ($F_1 = 0.854$) in identifying accurate semantic predications. Among five knowledge graph completion models, TransE outperformed others (MR = 0.923, Hits@1 = 0.417). Some known drugs linked to COVID-19 in the literature were identified, as well

Using Neo4j to start your Graph journey

What is Neo4j?

- Commercialised Graph Databases

What is Neo4j?

- Commercialised Graph Databases
- Fully integrated Graph Data Science Library as "low-code"

What is Neo4j?

- Commercialised Graph Databases
- Fully integrated Graph Data Science Library as "low-code"
- Graph visualisation tool

What is Neo4j?

- Commercialised Graph Databases
- Fully integrated Graph Data Science Library as "low-code"
- Graph visualisation tool
- It's Free!

What is Neo4j?

- Commercialised Graph Databases
- Fully integrated Graph Data Science Library as "low-code"
- Graph visualisation tool
- It's Free!

DB-ENGINES

#1 Most Popular Graph Database with Developers

☐ include secondary database models — 36 systems in ranking, November 2021

Rank Nov 2021	Rank Oct 2021	Rank Nov 2020	DBMS	Database Model	Score Nov 2021	Score Oct 2021	Score Nov 2020
1.	1.	1.	Neo4j	Graph	57.98	+0.11	+4.45
2.	2.	2.	Microsoft Azure Cosmos DB	Multi-model	40.82	+0.54	+8.32
3.	↑ 4.	3.	ArangoDB	Multi-model	5.10	+0.65	-0.27
4.	↓ 3.	↑ 5.	Virtuoso	Multi-model	4.81	+0.12	+2.28
5.	5.	↓ 4.	OrientDB	Multi-model	4.64	+0.59	-0.66
6.	6.	↑ 8.	GraphDB	Multi-model	2.83	+0.18	+0.72
7.	↑ 8.	↓ 6.	Amazon Neptune	Multi-model	2.60	+0.21	+0.17
8.	↓ 7.	↓ 7.	JanusGraph	Graph	2.54	+0.02	+0.17
9.	9.	↑ 13.	TigerGraph	Graph	2.02	+0.04	+0.89
10.	10.	↑ 11.	Stardog	Multi-model	1.97	+0.04	+0.51

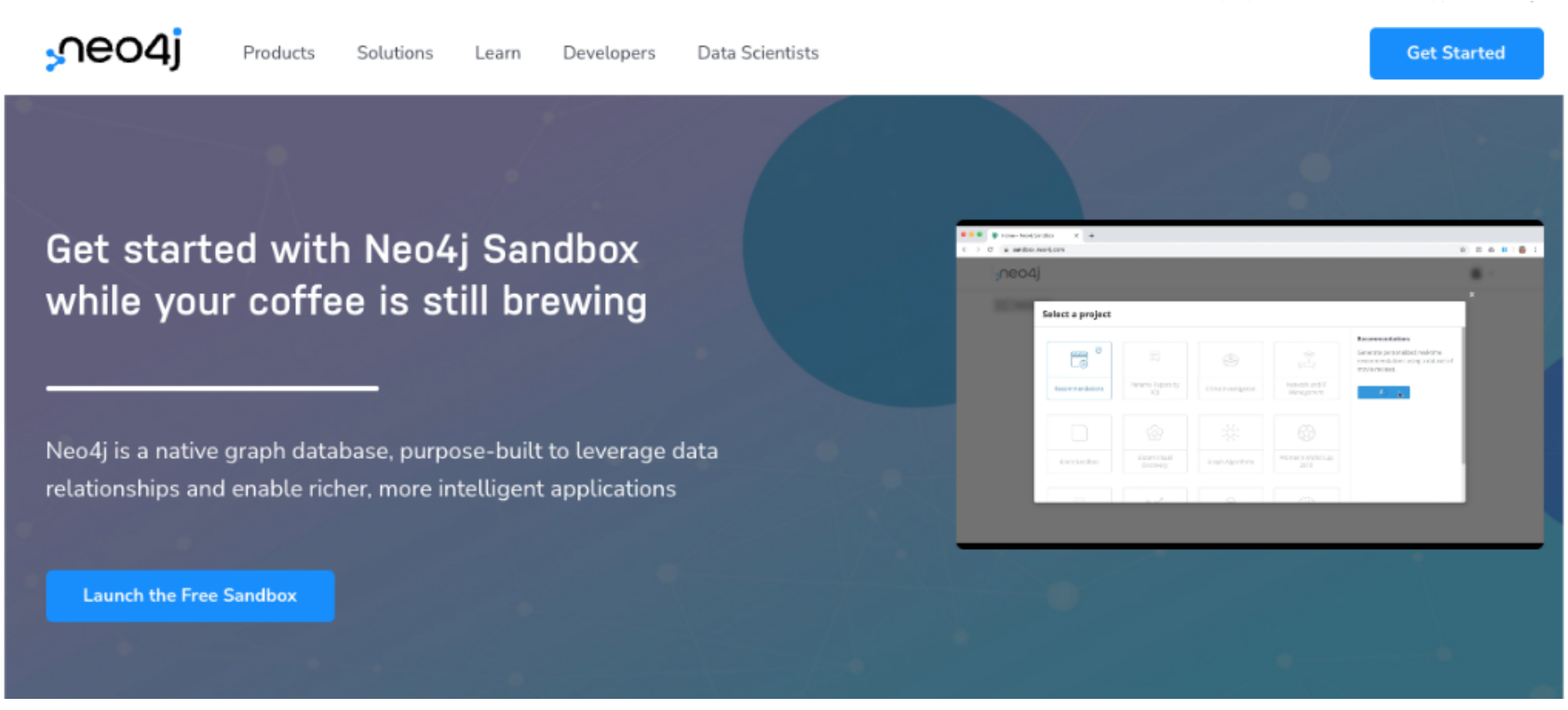
TOP 20 Database

200k+ Developers

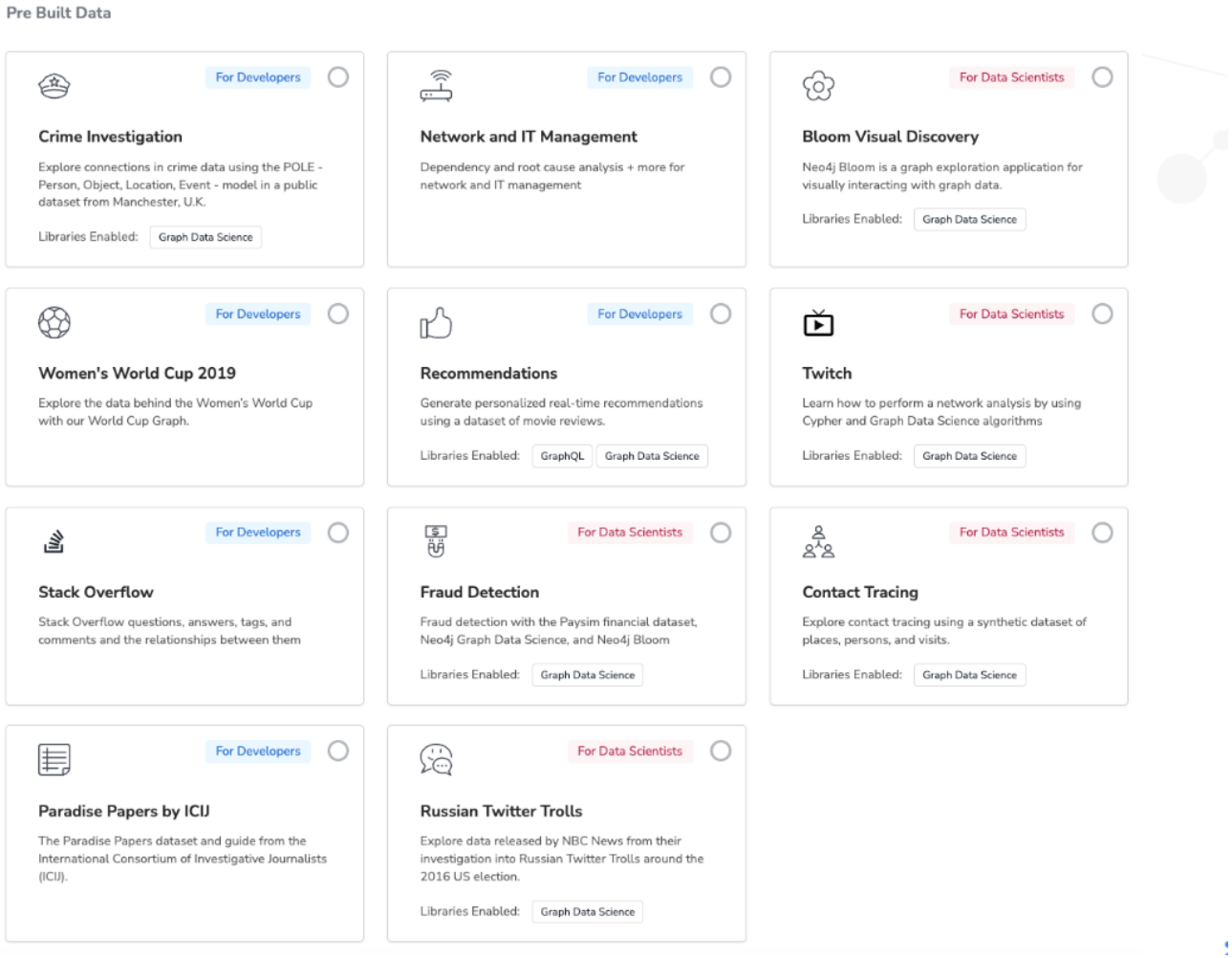
72k+ Meetup Members Globally

50k+ Members with LinkedIn Skills

Neo4j Sandbox



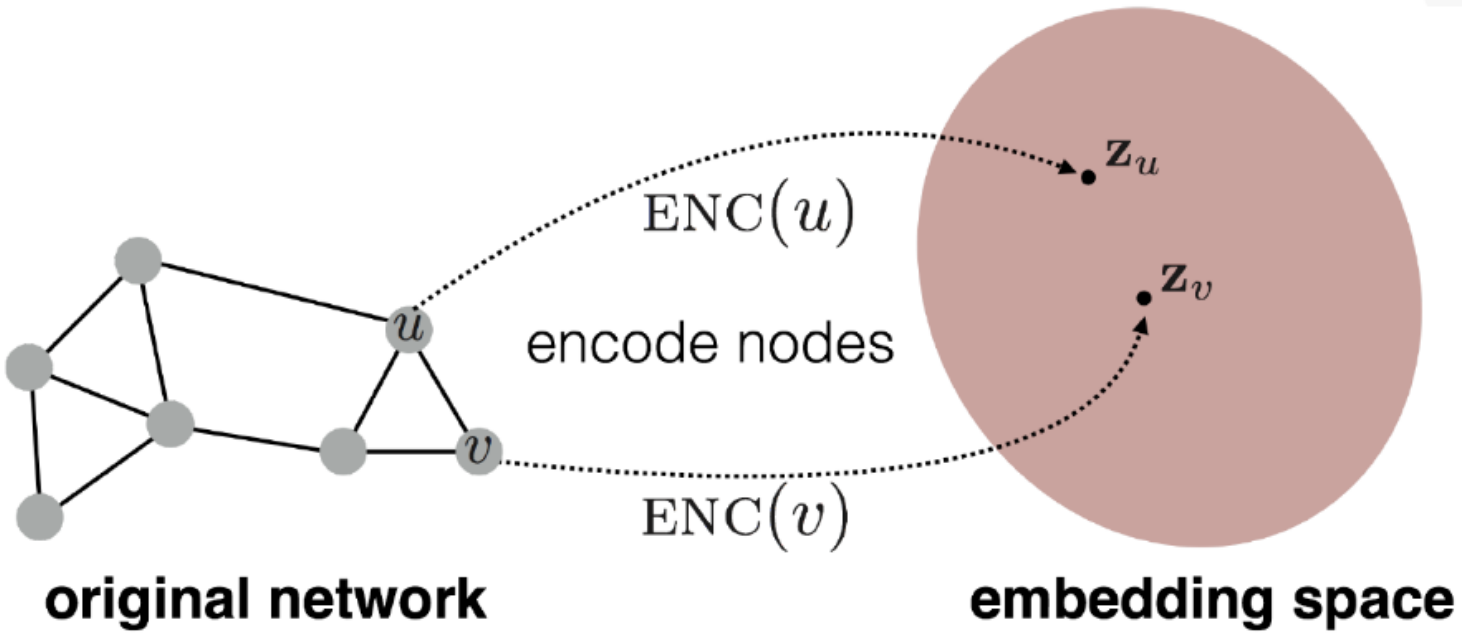
Neo4j Sandbox



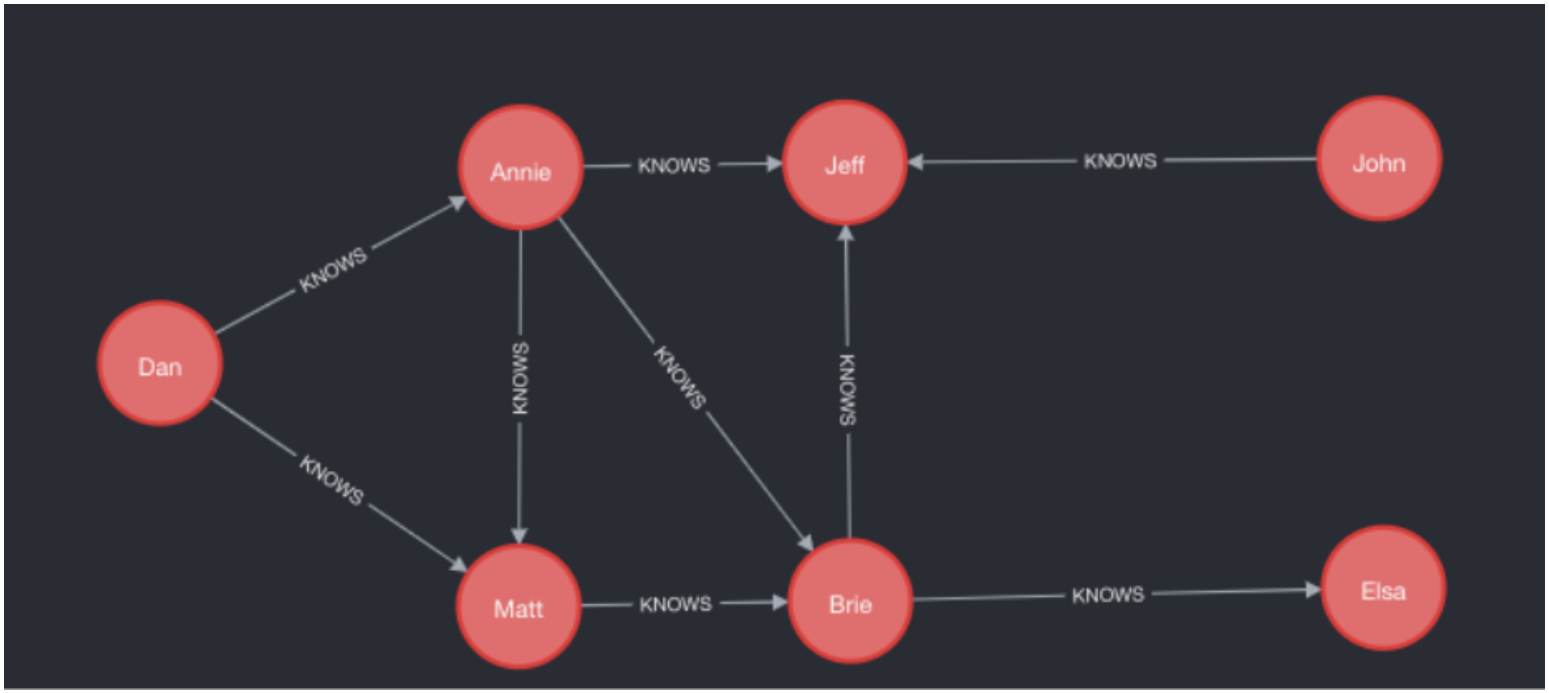
Node Embedding

What are Node embeddings?

The representation of nodes as low-dimensional vectors that summarize their graph position, the structure of their local graph neighborhood as well as any possible node features.



Node Embedding using a Toy example



Loading in Neo4j

```
CREATE
  (dan:Person {name: 'Dan', age: 18}),
  (annie:Person {name: 'Annie', age: 12}),
  (matt:Person {name: 'Matt', age: 22}),
  (jeff:Person {name: 'Jeff', age: 51}),
  (brie:Person {name: 'Brie', age: 45}),
  (elsa:Person {name: 'Elsa', age: 65}),
  (john:Person {name: 'John', age: 64}),

  (dan)-[:KNOWS {weight: 1.0}]->(annie),
  (dan)-[:KNOWS {weight: 1.0}]->(matt),
  (annie)-[:KNOWS {weight: 1.0}]->(matt),
  (annie)-[:KNOWS {weight: 1.0}]->(jeff),
  (annie)-[:KNOWS {weight: 1.0}]->(brie),
  (matt)-[:KNOWS {weight: 3.5}]->(brie),
  (brie)-[:KNOWS {weight: 1.0}]->(elsa),
  (brie)-[:KNOWS {weight: 2.0}]->(jeff),
  (john)-[:KNOWS {weight: 1.0}]->(jeff);
```

Node Embedding - using a Toy example

Step 2: Project the Graph:

```
CALL gds.graph.create(
'persons',
'Person',
KNOWS:
orientation: 'UNDIRECTED'
)
```

Step 3: Write Basic FastRp embeddings:

```
CALL gds.fastRP.write(
'persons',
embeddingDimension: 2,
writeProperty: 'FastRP-embed',
randomSeed: 42
)
YIELD nodePropertiesWritten
```

Node Embedding

entity	FastRP-embed
"Dan"	[0.7539142966270447, -1.6569727659225464]
"Annie"	[1.3116023540496826, -1.1916708946228027]
"Matt"	[0.9809403419494629, -1.7251838445663452]
"Jeff"	[1.3046178817749023, -1.311225414276123]
"Brie"	[1.3702197074890137, -1.4258880615234375]
"Elsa"	[1.5773502588272095, -0.8164965510368347]

Node Embedding - finding similarities

Step 4: Run basic Cosine similarity:
MATCH (p1:Person)
MATCH (p2:Person)
WHERE id(p2) $>$ id(p1)
RETURN p1.name as from, p2.name as to,
gds.alpha.similarity.cosine(p1.‘FastRP-embed‘, p2.‘FastRP-embed‘)
AS similarity
ORDER BY similarity DESC

Node Embedding

	from	to	similarity
1	"Jeff"	"Brie"	0.9998489604880746
2	"Annie"	"Jeff"	0.9987302286144143
3	"Annie"	"Brie"	0.9977038252612142
4	"Dan"	"Matt"	0.9959516923877233
5	"Elsa"	"John"	0.9938025144088152

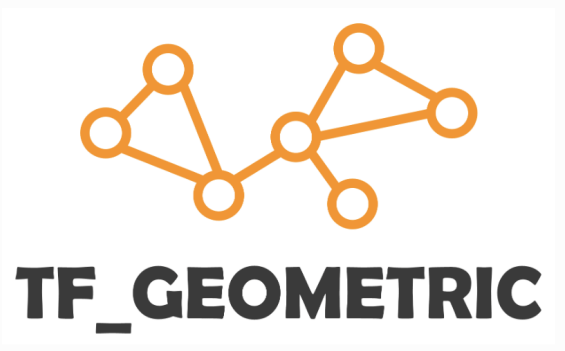
Node Embedding

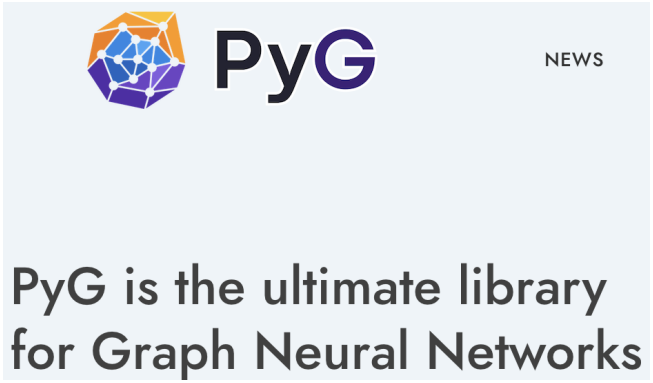
17	"Dan"	"Annie"	0.9186006881331586
18	"Matt"	"John"	0.8939398357000139
19	"Dan"	"John"	0.8500333287906242
20	"Matt"	"Elsa"	0.8385791578380498
21	"Dan"	"Elsa"	0.7862140346462471

Other software packages

Packages


DEEP
GRAPH
LIBRARY


TF_GEOMETRIC


PyG
NEWS
PyG is the ultimate library
for Graph Neural Networks

Thank you!

- A special thanks to Prof. George Magoulas, Dr. Petar Veličković

Links, resources and contact

- cneys01@dcs.bbk.ac.uk
- https://github.com/Kristof-Neys
- articles: https://kristof-neys-58246.medium.com/
- Best book: Graph representation learning, Will Hamilton.
  Free online: https://www.cs.mcgill.ca/~wlh/grl_book/
- others:
  Petar Veličković: https://petar-v.com/,
  Michael Bronstein:
  https://medium.com/@michael.bronstein