Graph Based AI

Mark Needham, Developer Relations at Neo4j

neo4j

Agenda

- High Level Overview
- Context for AI
- Graph Examples

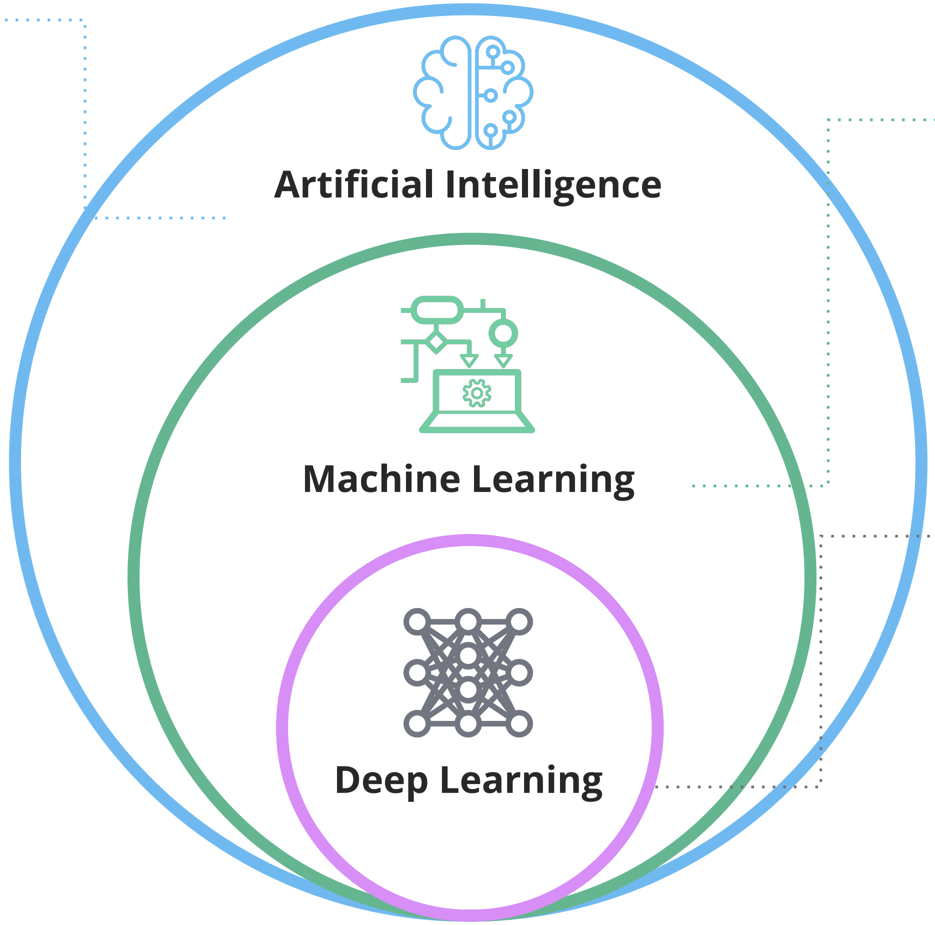
Machine Intelligence

AI consists of several subsets of technologies that solve problems in different ways

Artificial Intelligence (AI)

A computer process that has learned to solve tasks in a way that mimics human decisions

AI solutions today are mostly used for very specific tasks, versus general applications



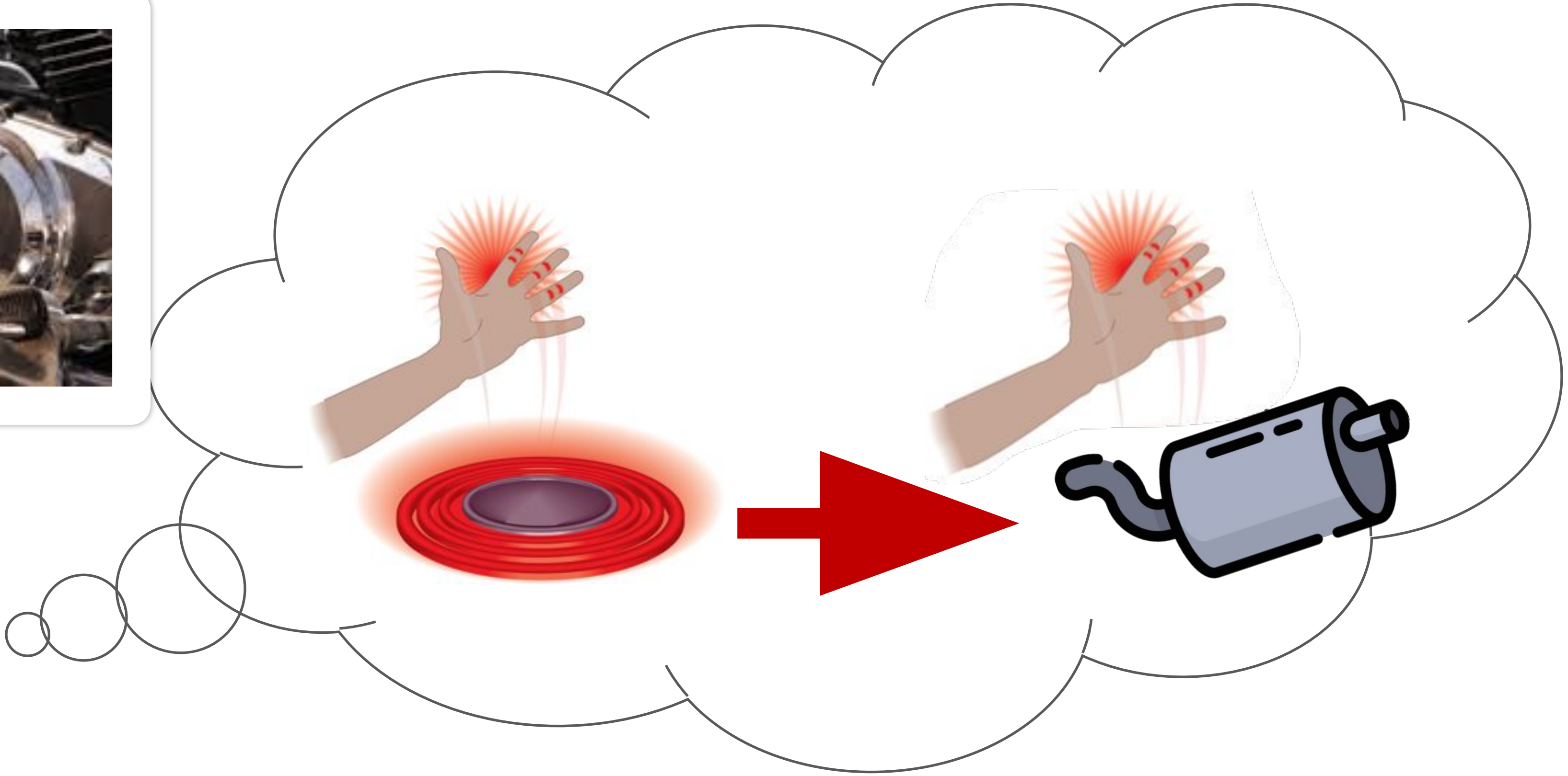
Machine Learning (ML)

Uses algorithms to help computers learn by task specific examples and progressive improvements, without explicit programming

Deep Learning (DL)

Uses a cascade of processing layers modeled on neural network to learn data representations such as features or classifications

vs.

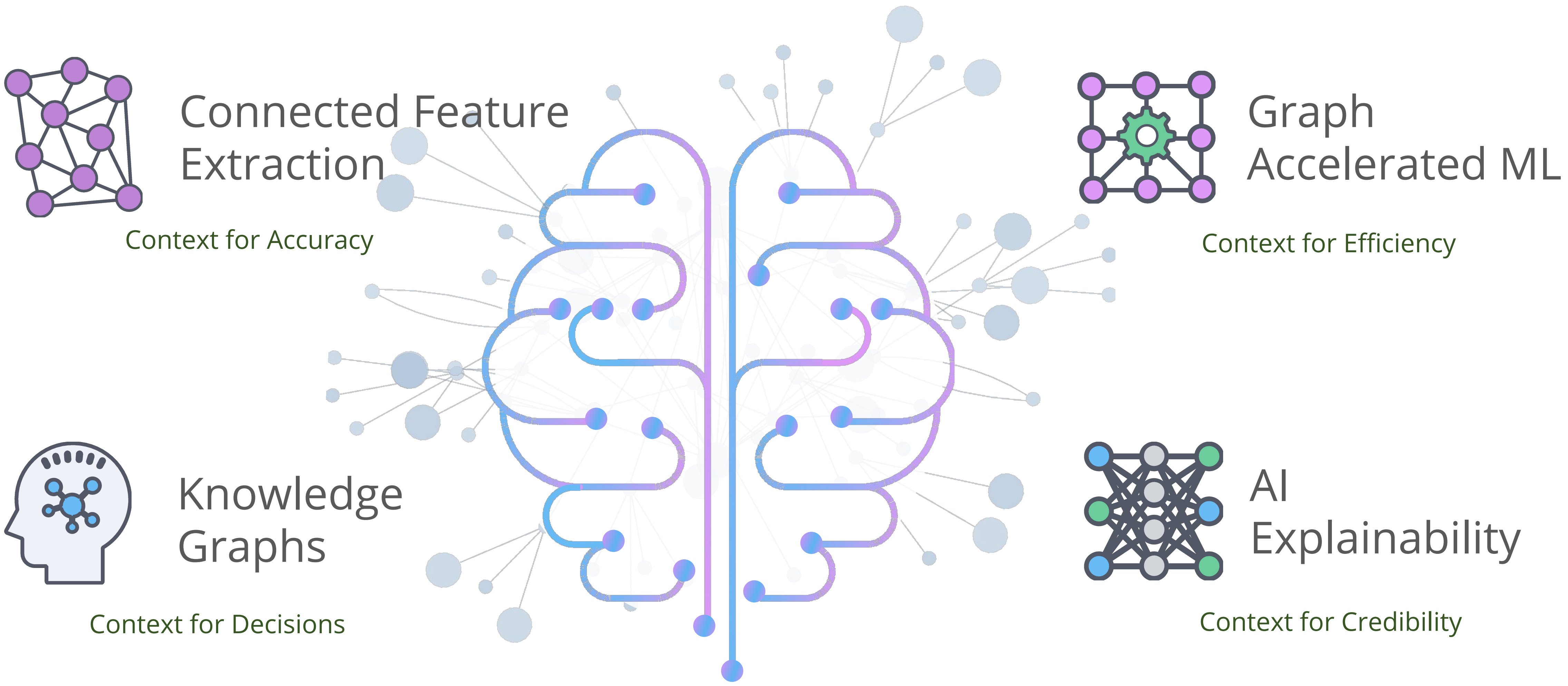


Contextual Learning

But We’re Not There Yet…

- Missing contextual information
  - Relationships
  - Adjacent knowledge
- Layers of complexity without using context to optimize processes
- Missing context of the right data for the right decision
- Unexplainable without the context for how decisions are made

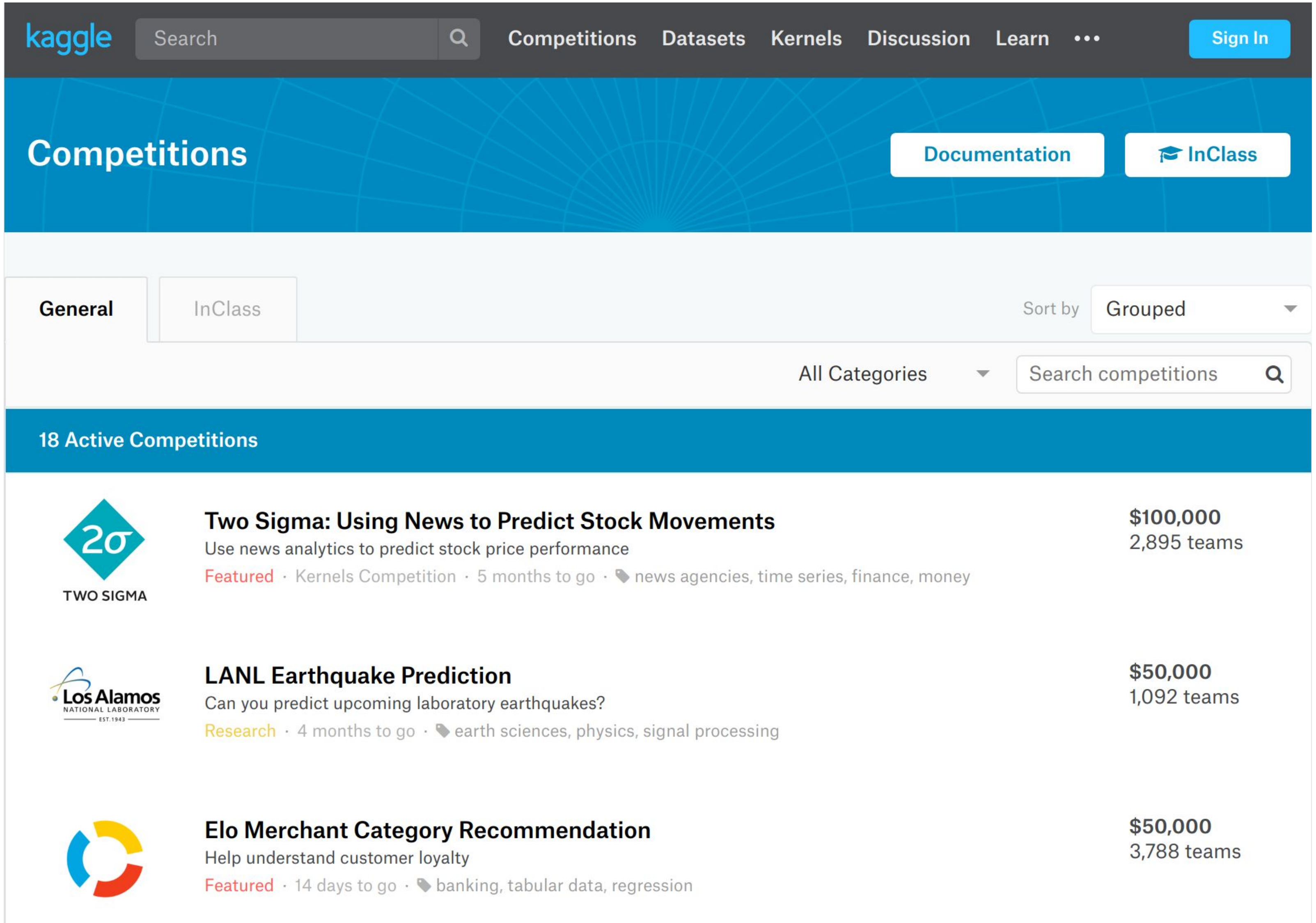
Graphs Enhance AI by Providing Context



Graphs are Context for Accuracy

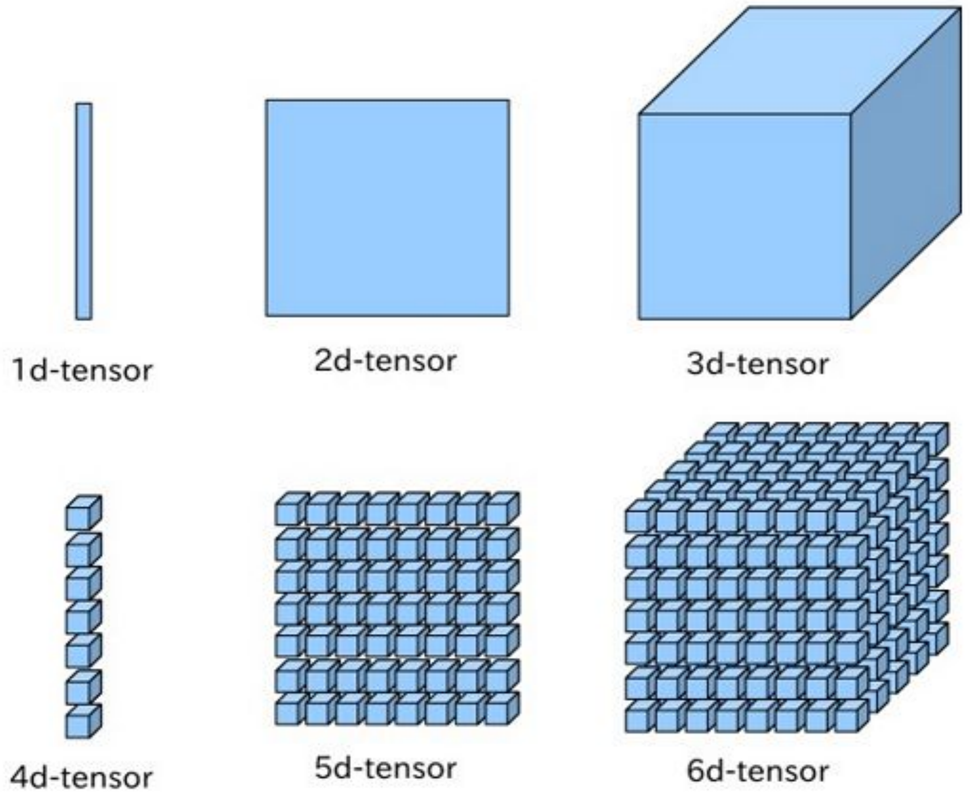
Connected Feature Extraction

Kaggle taught us that data is flat...

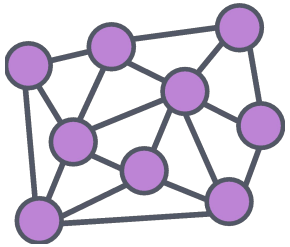


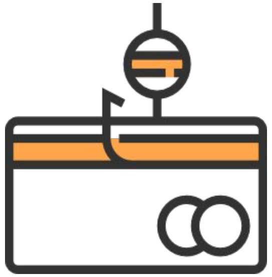
…but sometimes it's who you know

- Relationships are often the strongest predictors of behavior
- Current machine learning methods rely on vectors, matrices, and tensors built from tables
- These methods simplify, or leave out entirely, predictive relationship and network data
- Graphs add highly predictive features to these models, adding accuracy without altering algorithms
- Graphs can infer relationships and add data where sparse



Connected Feature Extraction for Predictive Lift





Methods:

- Engineered features (labeled and inferred relationships)

- Graph algorithms (e.g. Centrality and Community Detection)

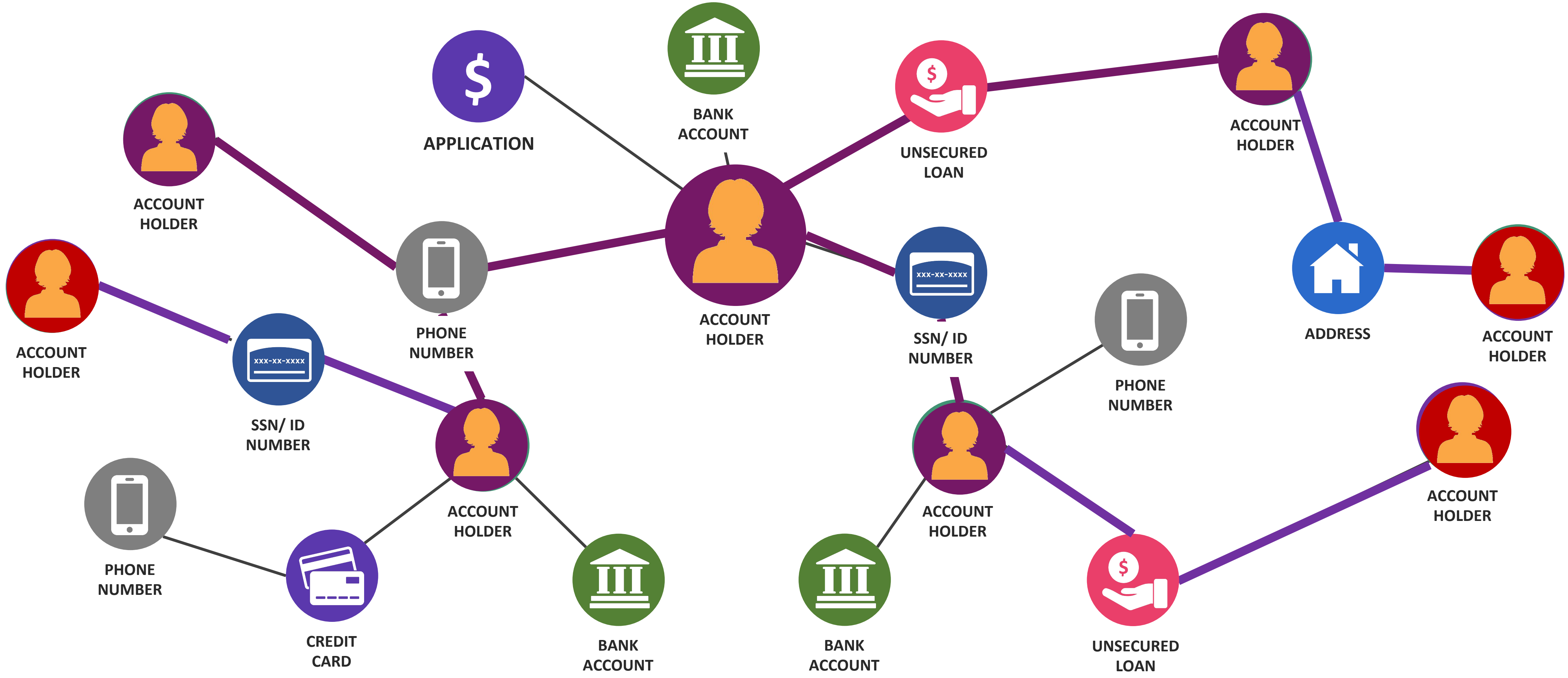
- Graph embeddings (DeepWalk, DeepGL, Node2Vec, etc.)

Example: Financial Crimes

- Transaction Fraud
- Anti-money laundering (AML)
- Claims Fraud
- Credit Fraud
- Compliance and investigation

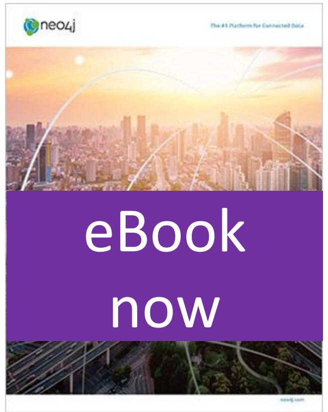
Connected Feature Extraction for Fraud Prevention

4 – 2 Hop Connections
3 – 4 Hop Connections



Graph Algorithms in Neo4j

GraphConnect 2017 → **GraphConnect 2018**



Pathfinding & Search

GraphConnect 2017:
- Parallel Breadth First Search & DFS
- Shortest Path
- Single-Source Shortest Path
- All Pairs Shortest Path
- Minimum Spanning Tree

GraphConnect 2018:
- A* Shortest Path
- Yen's K Shortest Path
- K-Spanning Tree (MST)

Centrality / Importance

GraphConnect 2017:
- Degree Centrality
- Closeness Centrality
- Betweenness Centrality
- PageRank

GraphConnect 2018:
- Harmonic Closeness Centrality
- Dangalchev Closeness Centrality
- Wasserman & Faust Closeness Centrality
- Approximate Betweenness Centrality
- Personalized PageRank

Community Detection

GraphConnect 2017:
- Triangle Count
- Clustering Coefficients
- Connected Components (Union Find)
- Strongly Connected Components
- Label Propagation
- Louvain Modularity – 1 Step

GraphConnect 2018:
- Balanced Triad (identification)
- Louvain – Multi-Step

O'REILLY

2019 Q1

Similarity & ML Workflow

GraphConnect 2018:
- **Euclidean Distance**
- **Cosine Similarity**
- **Jaccard Similarity**
- **Random Walk**
- **One Hot Encoding**

Reference Implementations for Graph Embeddings (Node to Vector)
- DeepGL
- DeepWalk

Link Prediction

The Link Prediction Problem for Social Networks*

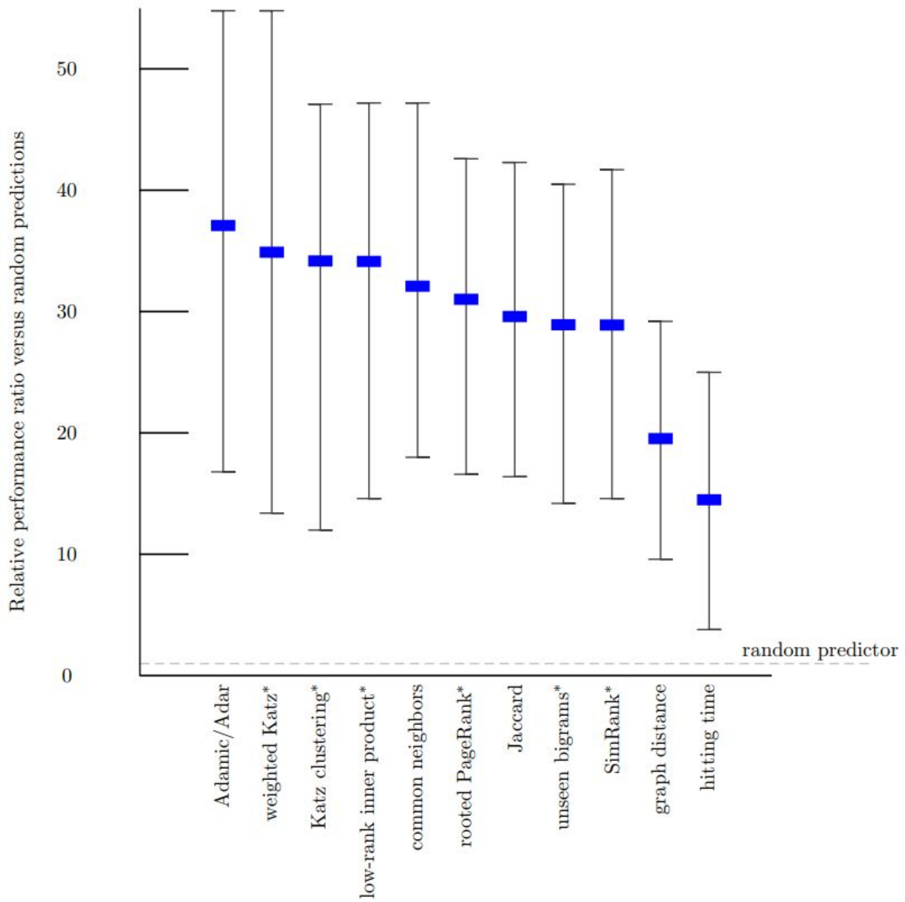
David Liben-Nowell†
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139 USA
dln@theory.lcs.mit.edu

Jon Kleinberg‡
Department of Computer Science
Cornell University
Ithaca, NY 14853 USA
kleinber@cs.cornell.edu

January 8, 2004

Abstract

Given a snapshot of a social network, can we infer which new interactions among its members are likely to occur in the near future? We formalize this question as the *link prediction problem*, and develop approaches to link prediction based on measures for analyzing the "proximity" of nodes in a network. Experiments on large co-authorship networks suggest that information about future interactions can be extracted from network topology alone, and that fairly subtle measures for detecting node proximity can outperform more direct measures.



Graph Embeddings

Deep Feature Learning for Graphs

Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed

Abstract—This paper presents a general graph representation learning framework called DeepGL for learning deep node *and* edge representations from large (attributed) graphs. In particular, DeepGL begins by deriving a set of base features (*e.g.*, graphlet features) and automatically learns a multi-layered hierarchical graph representation where each successive layer leverages the output from the previous layer to learn features of a higher-order. Contrary to previous work, DeepGL learns *relational functions* (each representing a feature) that generalize across-networks and therefore useful for graph-based transfer learning tasks. Moreover, DeepGL naturally supports attributed graphs, learns interpretable graph representations, and is space-efficient (by learning sparse feature vectors). In addition, DeepGL is expressive, flexible with many interchangeable components, efficient with a time complexity of $\mathcal{O}(|E|)$, and scalable for large networks via an efficient parallel implementation. Compared with the state-of-the-art method, DeepGL is (1) **effective** for across-network transfer learning tasks *and* attributed graph representation learning, (2) **space-efficient** requiring up to $6\times$ less memory, (3) **fast** with up to $182\times$ speedup in runtime performance, and (4) **accurate** with an average improvement of 20% or more on many learning tasks.

Index Terms—Graph feature learning, graph representation learning, deep graph features, relational functions, higher-order features,

DeepWalk: Online Learning of Social Representations

Bryan Perozzi
Stony Brook University
Department of Computer Science

Rami Al-Rfou
Stony Brook University
Department of Computer Science

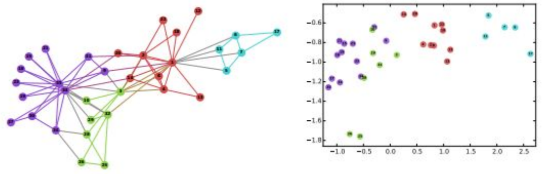
Steven Skiena
Stony Brook University
Department of Computer Science

{bperozzi, ralrfou, skiena}@cs.stonybrook.edu

ABSTRACT

We present DeepWalk, a novel approach for learning latent representations of vertices in a network. These latent representations encode social relations in a continuous vector space, which is easily exploited by statistical models. DeepWalk generalizes recent advancements in language modeling and unsupervised feature learning (or *deep learning*) from sequences of words to graphs.

DeepWalk uses local information obtained from trun-



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

Inductive Representation Learning on Large Graphs

William L. Hamilton[*]
wleif@stanford.edu

Rex Ying[*]
rexying@stanford.edu

Jure Leskovec
jure@cs.stanford.edu

Department of Computer Science
Stanford University
Stanford, CA, 94305

Abstract

Low-dimensional embeddings of nodes in large graphs have proved extremely useful in a variety of prediction tasks, from content recommendation to identifying protein functions. However, most existing approaches require that all nodes in the graph are present during training of the embeddings; these previous approaches are inherently *transductive* and do not naturally generalize to unseen nodes. Here we present GraphSAGE, a general *inductive* framework that leverages node feature

struc2vec: Learning Node Representations from Structural Identity

Leonardo F. R. Ribeiro
Federal University of Rio de Janeiro
Systems Eng. and Comp. Science Dep.
leo@land.ufrj.br

Pedro H. P. Saverese
Federal University of Rio de Janeiro
Systems Eng. and Comp. Science Dep.
savarese@land.ufrj.br

Daniel R. Figueiredo
Federal University of Rio de Janeiro
Systems Eng. and Comp. Science Dep.
daniel@land.ufrj.br

ABSTRACT

Structural identity is a concept of symmetry in which network nodes are identified according to the network structure and their relationship to other nodes. Structural identity has been studied in theory and practice over the past decades, but only recently has it been addressed with representational learning techniques. This work presents *struc2vec*, a novel and flexible framework for learning latent representations for the structural identity of nodes. *struc2vec* uses a hierarchy to measure node similarity at different scales, and constructs a multilayer graph to encode structural similarities and generate structural context for nodes. Numerical

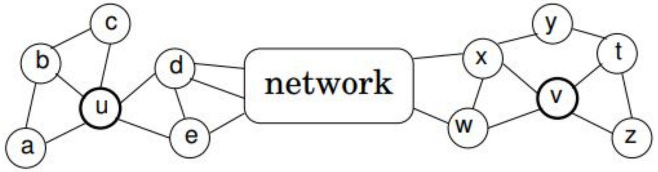
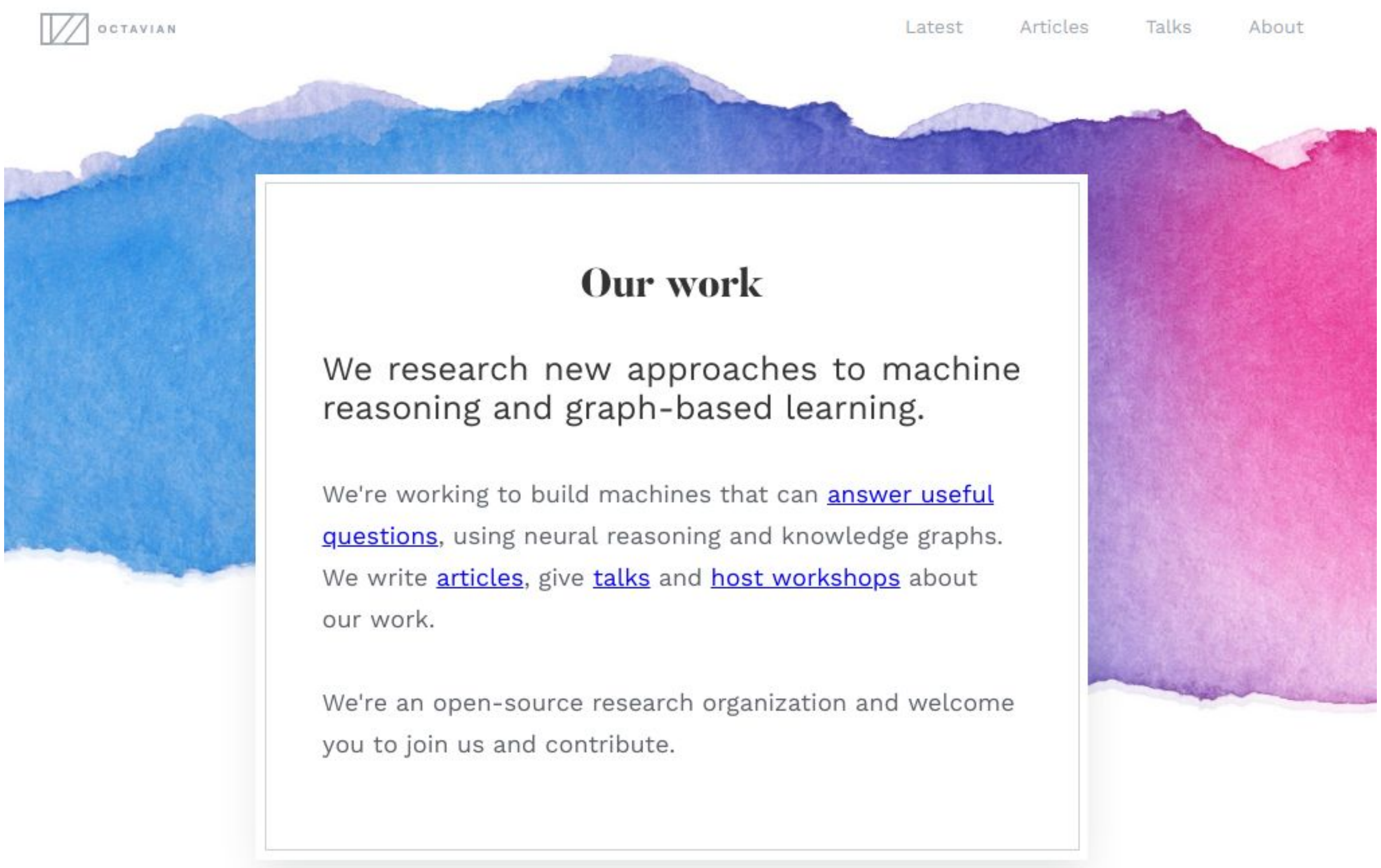
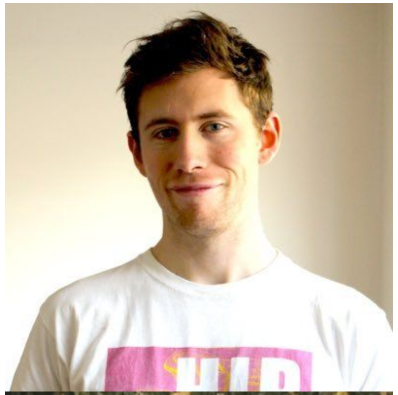


Figure 1: An example of two nodes (u and v) that are structurally similar (degrees 5 and 4, connected to 3 and 2 triangles, connected to the rest of the network by two nodes), but very far apart in the network.

And if we don't want to flatten our graphs?





And if we don't want to flatten our graphs?

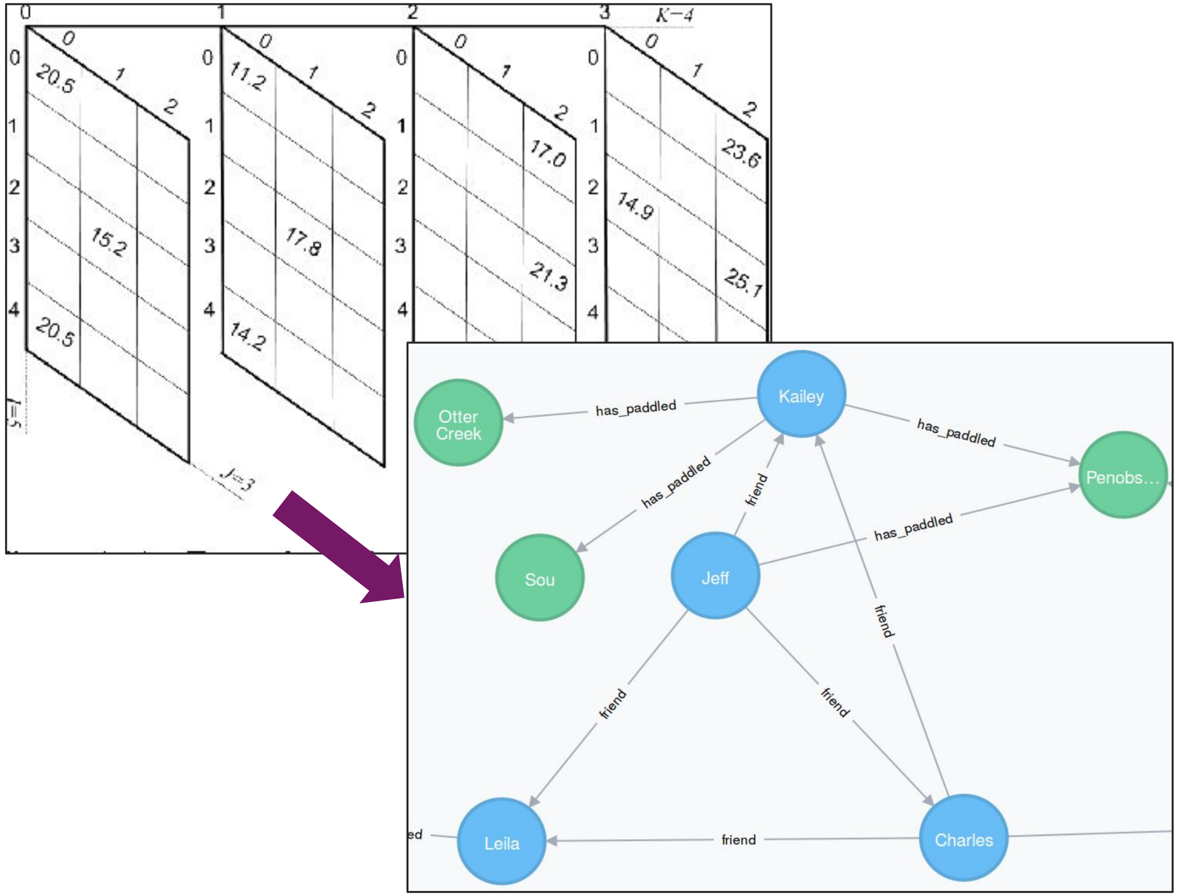
	Model	Learned Params	Data
Linear Regression	y = mx + c	m	List<(x, y)>
Image Classifier	P(class | image) = CNN(image)	CNN weights	List<(image, class)>
NN Embedding	P(path | node) = NN(path)	node embedding, NN weights	Node dictionary List<(path, node)>
Graph Regression	node.prop = F([sub]graph)	F weights	List<(node, [sub]graph)> *or* Graph, pattern
Graph Classifier	P(class | [sub]graph) = F([sub]graph)	F weights	List<([sub]graph, class)> *or* Graph, pattern
Graph Embedder	P(subgraph | node) = F(graph)	node embedding, F weights	Graph

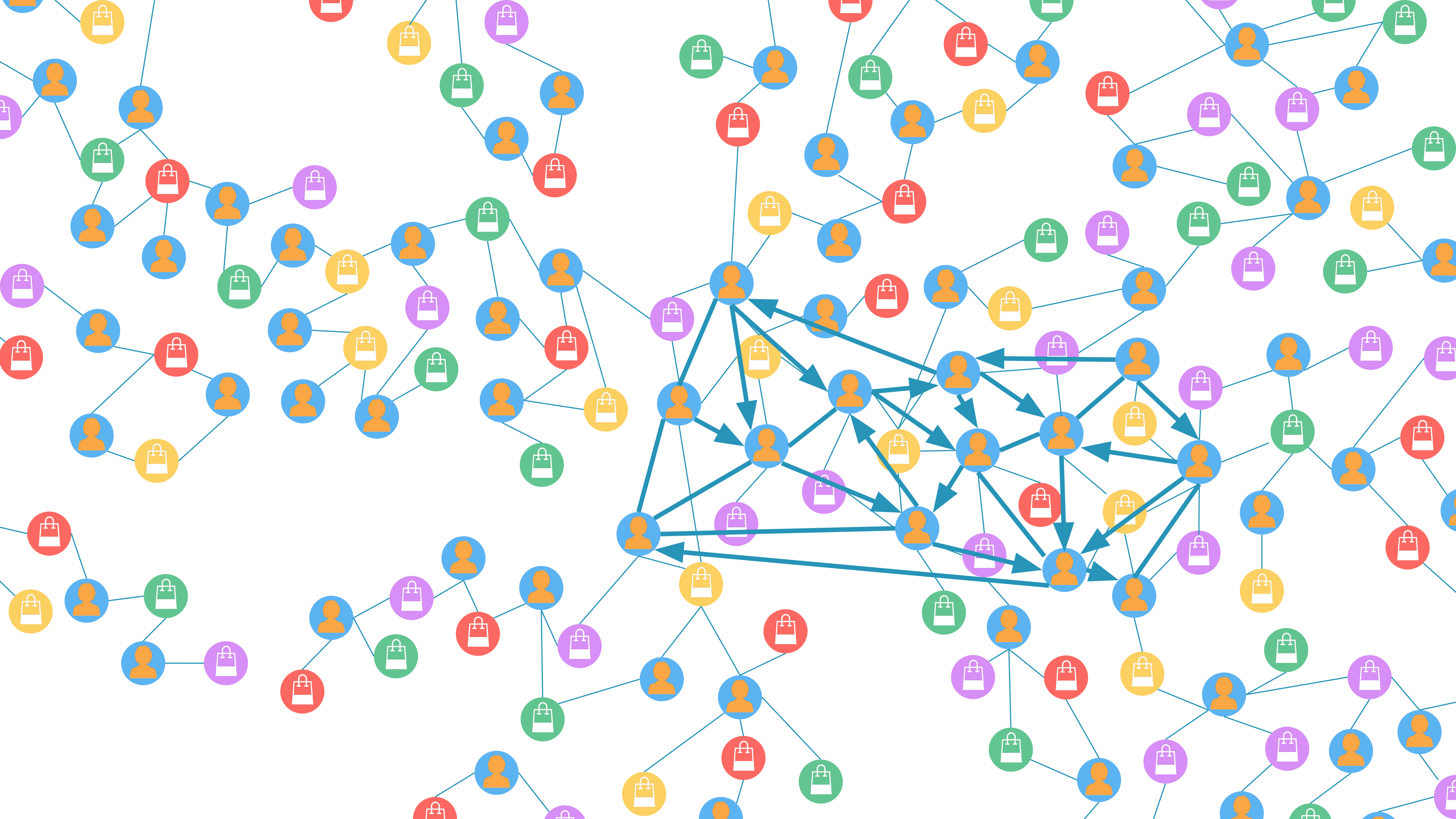
Graphs are Context for Efficiency

Graph Accelerated ML

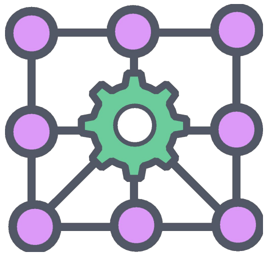
Brute force is as inelegant as it sounds

- 56% of enterprise CIOs say iterative model training is the largest ML challenge[1]
- Renting more and more GPU time is not the answer – not every problem is "embarrassingly parallel"
- Table joins bog down data pipelines





Accelerate Your ML Process



Methods:

- Replace table joins with graph queries
- Replace sparse matrices and directional relationships with more efficient graph structures (i.e. collaborative filtering via Cypher query vs. matrix factorization)
- Use subgraph filtering to accelerate ML pipelines (Cypher queries, collaborative filtering, community detection, clustering, etc.)

Example: Recommendations

- Real time recommendations
- Customer Segmentation/KYC
- Churn analysis
- Dynamic pricing
- Promotions
- Patient modeling

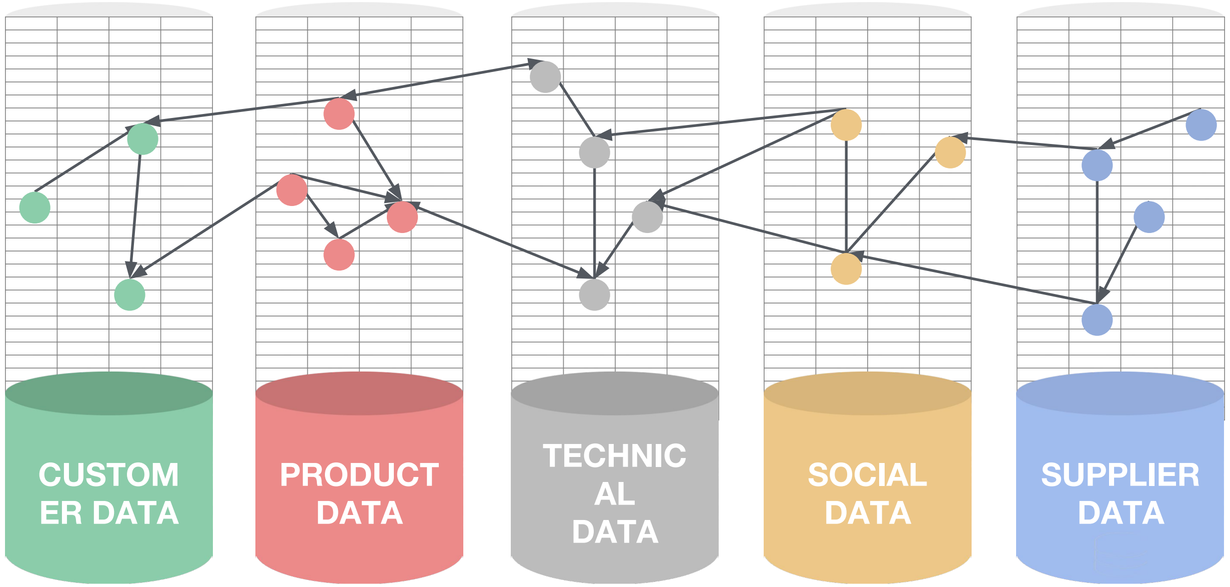
Graphs are Context for Decisions

Knowledge Graphs

neo4j

Context doesn’t fit cleanly in an equation

- A connected, dynamic, and understandable repository of different data types
- Link siloed or external data sources in an intelligent way
- Key to understand your unique, enterprise language
- **Knowledge Base ≠ Knowledge Graph**



3 Types of Knowledge Graphs

Context Rich Search

Internal knowledge documents & files, with meta data tagging

Examples:
- Search
- Customer support
- Document classification

External Insight Sensing

External data source aggregation mapped to entities of interest

Examples:
- Supply chain/compliance risk
- Market activity aggregation
- Sales opportunities

THOMSON REUTERS

Enterprise NLP

Graph technical terms, acronyms, abbreviations, misspellings, etc.

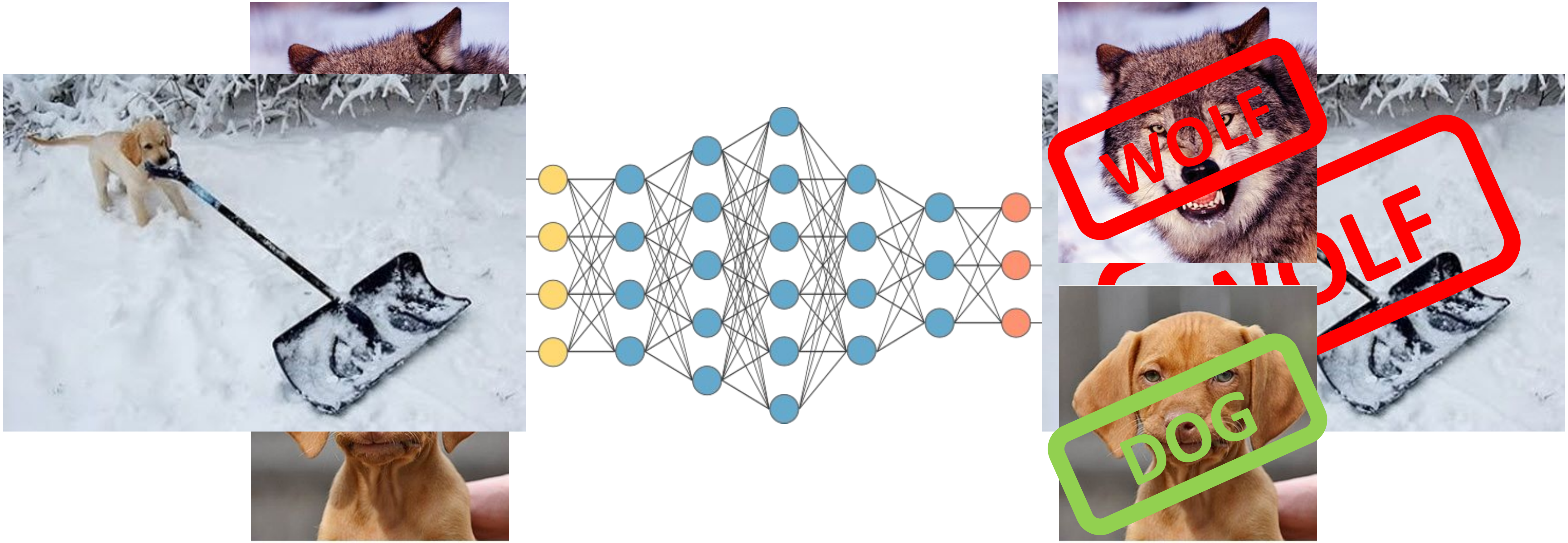
Examples:
- Improved search
- Chatbot implementation
- Improved classification

Graphs are Context for Credibility

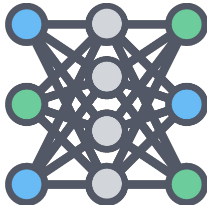
AI Explainability

neo4j

Credibility Matters

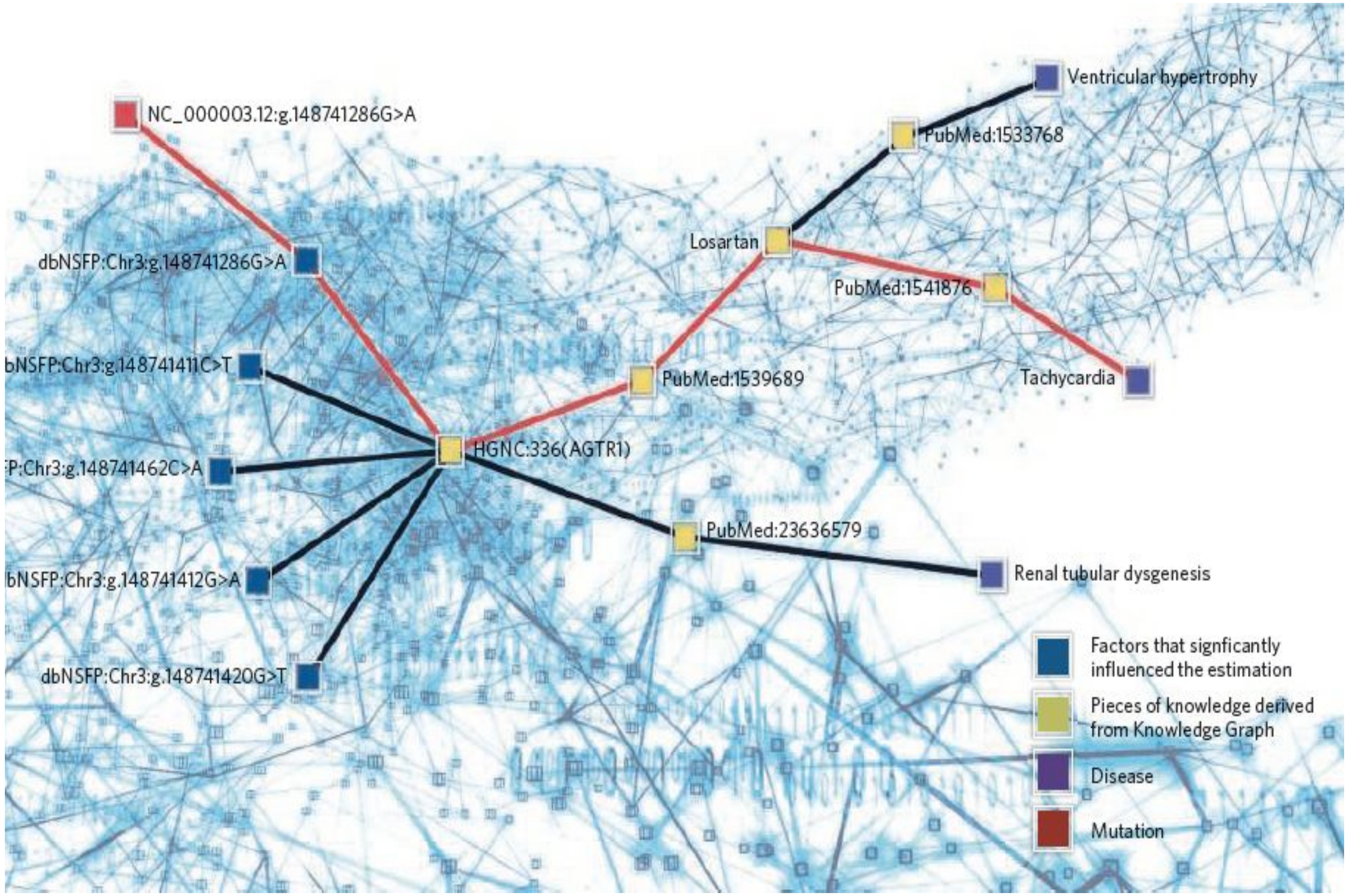


When you need to open the Black Box



- There are multiple types of explainability
  - Explainable data – what data was used to train the model, and why?
  - Explainable predictions – what features and weights were used for this particular prediction?
  - Explainable algorithms – what are the individual layers and thresholds to a prediction?
- How do you add predictability, without reducing performance?

When you need to open the Black Box



Explainable Data: Graphs provide data lineage - when, where, and often why data was accessed

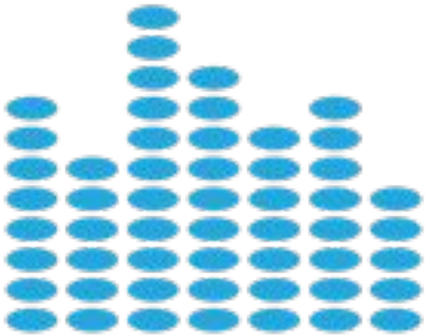
Explainable Predictions: Associating nodes in a neural network to a labelled knowledge graph allows for traversing related documents to an explanation

Explainable Algorithms: Early research shows that constructing tensors from graphs using weighted relationships may lead to explainable neural network algorithms

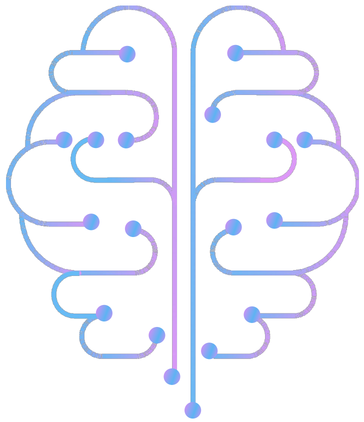
Graphs are Context for AI

neo4j

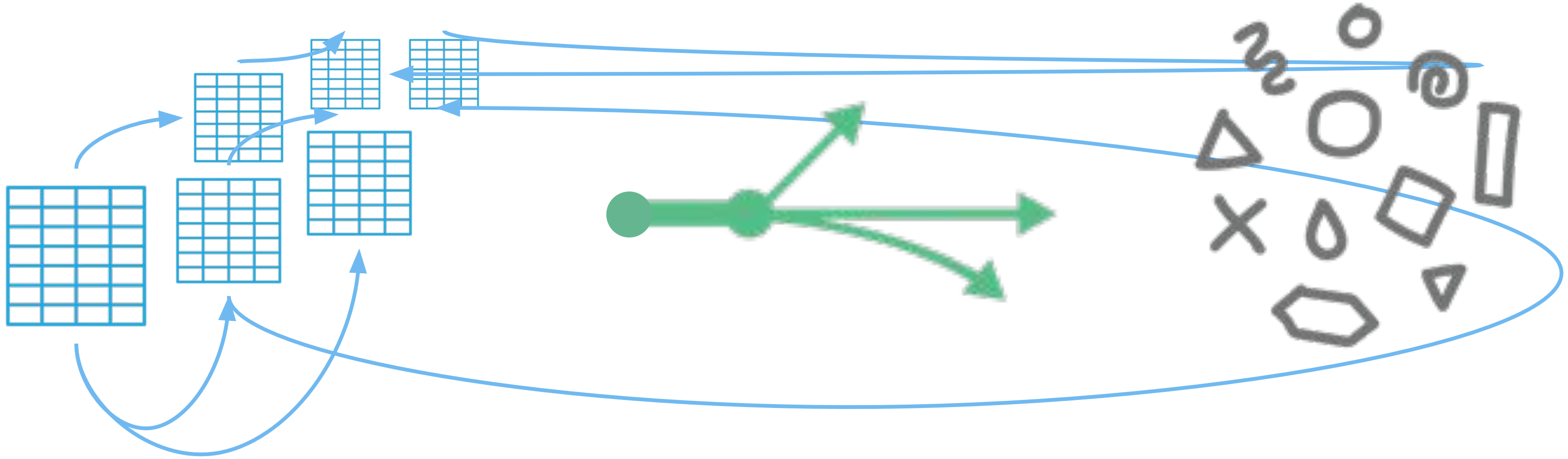
Aggregation is not Intelligence



Learning without Context is not Intelligence



Will Graphs Improve Your AI?



Predictive Relationships & Network Components

Table Joins

Sparse Matrices

Directional

Heterogenous Knowledge Sources

Thanks for listening!

Mark Needham

mark.needham@neo4j.com

@markhneedham

neo4j