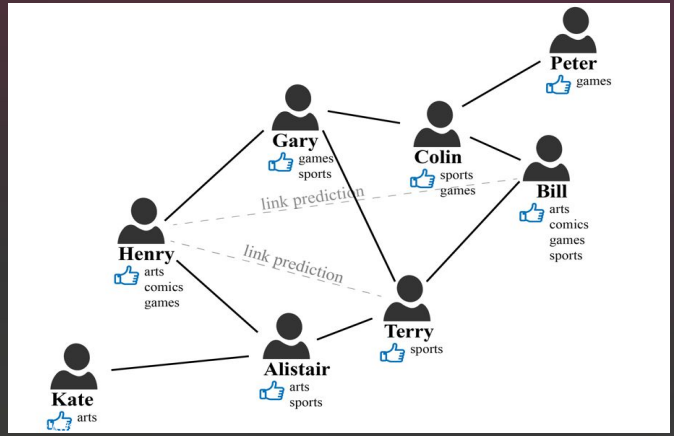
Querying Graphs and their Representations

Team Singh

Motivation

General Overview

- **What is a Graph Neural Network (GNN)?**
  - **Input:** Graph + feature vectors
  - **Output:** Graph embeddings
  - **Goal:** Learn and *predict*
- **Examples**:
  - Social Media Recommendations
  - Traffic Prediction



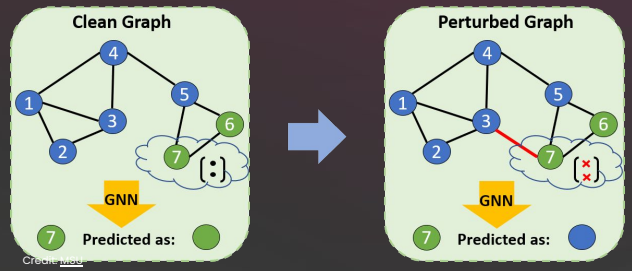
Problem Statement

We will develop an efficient framework capable of simultaneously comparing and analyzing large graphs across their topological and embedding spaces.

Currently, no such framework or query language exists.

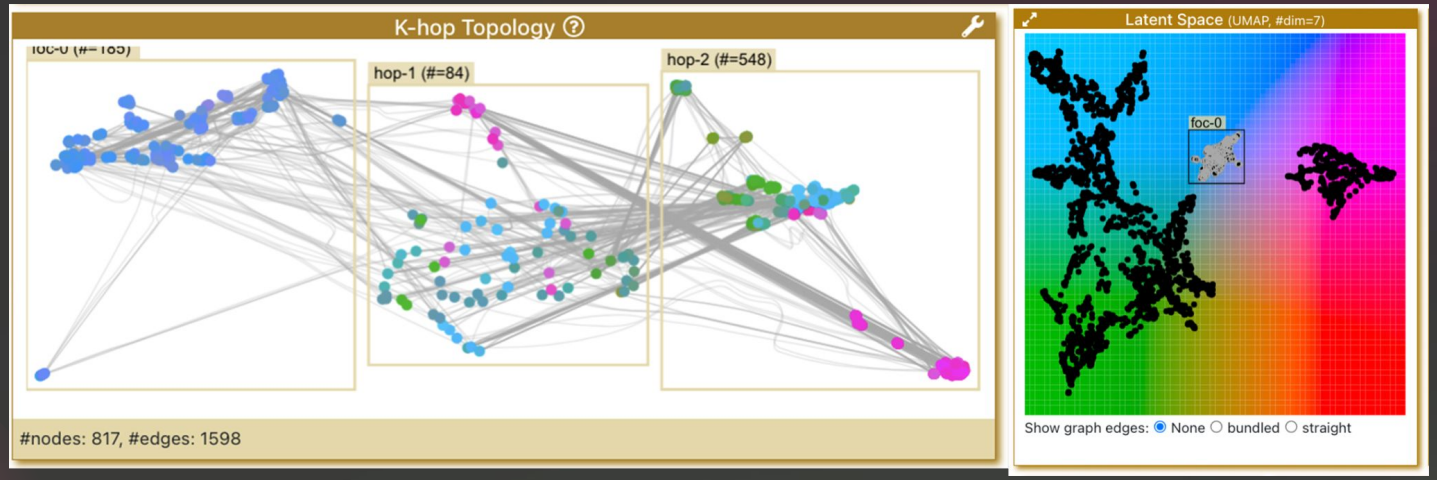
Motivation

- **What cannot be done?**
  - Large queries on graph data + embeddings
  - Data-based output
- **Where would this make an impact?**
  - Node Vulnerability Analysis
  - Feature Analysis
  - Graph Structure and Learning Outcomes
  - GNN Model Comparisons



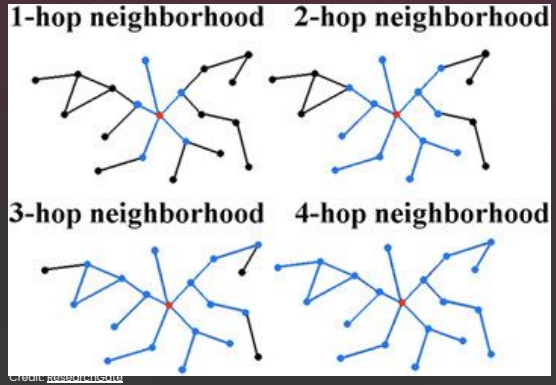
CorGIE

- (Left) ‘k’ steps (hops) away from a given node
- (Right) Shows the embeddings of a GNN



CorGIE

- **What is CorGIE missing?**
  - Maximum size of graph: 20,000 nodes
  - No querying function
  - No more than 2 ‘k-hops’
  - Lacks support for k-nearest neighbors
  - No longer actively maintained



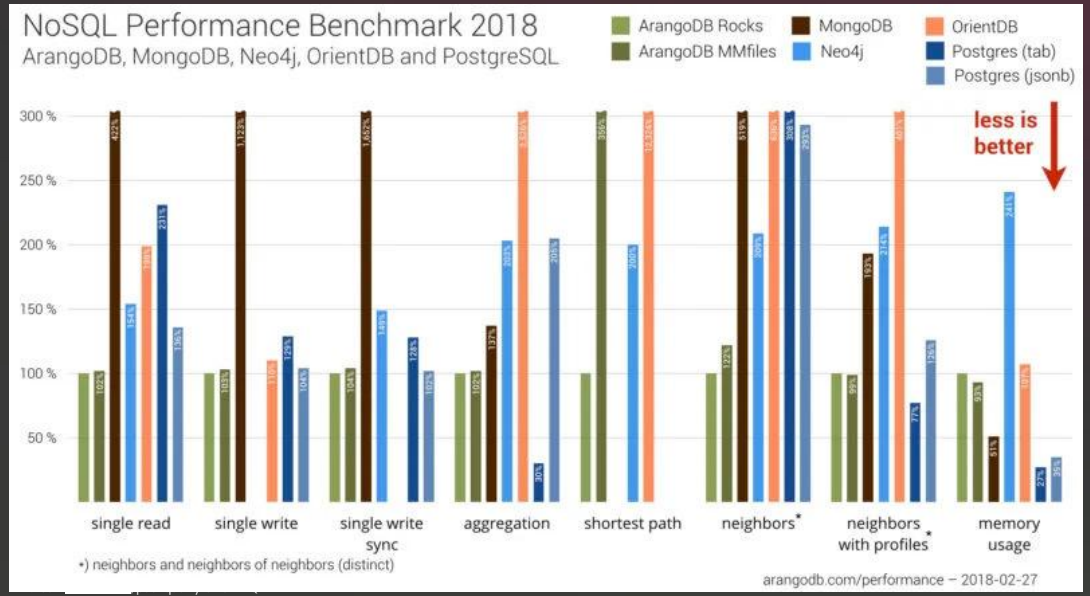
Proposed Solution + Future Plan

Graph Databases

- **Two most popular graph database (GraphDB) frameworks**
  - ArangoDB (AQL) and Neo4j (Cypher)
- **ArangoDB:**
  - Supports non-graph DB features (keys/values, docs, etc.)
  - Scales more efficiently
- **Neo4j:**
  - Has *in-house* APIs
  - Supports more *advanced* graph features

Graph Databases

- ArangoDB outperforms Neo4j in all use cases



Example AQL Query

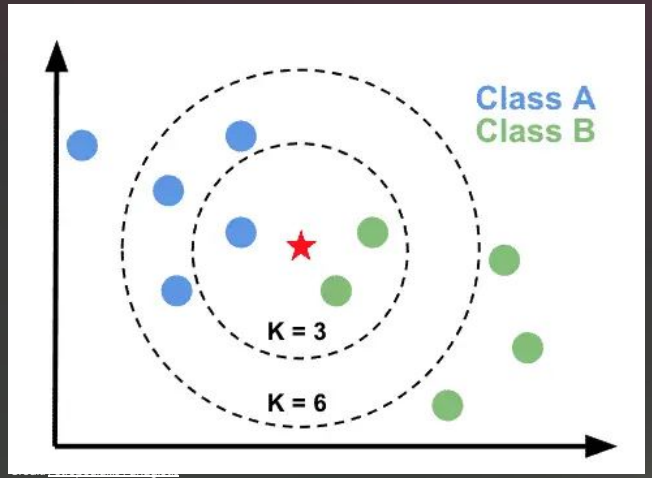
- Therefore, we will utilize ArangoDB and AQL
- Example query in AQL:

```
shortest_path_query = """FOR p IN OUTBOUND K_SHORTEST_PATHS 'Cities/Aberdeen'
  TO 'Cities/London'
  GRAPH 'RailNetwork'
      LIMIT 1
      RETURN {
          places: p.vertices[*]._key,
          travelTimes: p.edges[*].travel_time,
          travelTimeTotal: SUM(p.edges[*].travel_time)
      }"""
```

Credit: AQL Colab

Vector Similarity Search (FAISS)

- **What is vector similarity search?**
  - What is the distance between two vectors?
  - Closer vectors = more *similar*
- k-nearest neighbor search
  - Returns k most similar nodes
- **FAISS from Meta**
  - Python and C++ support
  - Speed / accuracy tradeoff



Example FAISS Query

- xq is a randomly-generated matrix with size (10000, 64)

4 nearest neighbors for each vector

```
k = 4
D, I = index.search(xq, k)
print("First five vectors:")
print(I[:5])
print()
print("Last five vectors:")
print(I[-5:])
```

xq: 0 - 4 and 9995-9999

```
First five vectors:
[[ 381  207  210  477]
 [ 526  911  142   72]
 [ 838  527 1290  425]
 [ 196  184  164  359]
 [ 526  377  120  425]]

Last five vectors:
[[ 9900 10500  9309  9831]
 [11055 10895 10812 11321]
 [11353 11103 10164  9787]
 [10571 10664 10632  9638]
 [ 9628  9554 10036  9582]]
```

Our Solution

- **Combine:**
  - GraphDB querying with AQL and ArangoDB
  - k-nearest neighbor queries on GNN embeddings with FAISS
- **SQL-analogous language:**
  - **Setup**: GraphDB, embedding matrix, FAISS function (optional)
  - **Input**: Well-formatted query
  - **Return**: an unordered dictionary

Our Solution

```
FOR researcher IN 'academics'
    FILTER researcher.department == 'computer science'
    FOR collaborator IN 1..3 OUTBOUND researcher GRAPH 'collaborationGraph'
    FILTER SIMILAR_TO(collaborator, 'researchInterests', researcher, 10)
    RETURN { researcherName: researcher.name,
             collaboratorName: collaborator.name }
```

What does this do?

- Finds potential research collaborators within 3-hops
- Verifies collaborator is within 10-most similar researchers

Evaluation

01

Functionality

Can it handle a wide range of queries?

02

Correctness

Is the output correct?

03

Scalability

Can it handle graphs of all sizes?

04

Efficiency

Can queries be returned in a reasonable time?

Implementation Plan

Implement functionality in C++ to run the queries

January

February

Design a query language, implement an API to connect the language to our functions

Expand project into other avenues and specific use cases of our language

March

April + May

Possibly write paper, else, continue expanding into other avenues

Wrap up research, begin to design final presentation and poster

June

References

[1]ArangoDB. AQL Example Queries on an Actors and Movies Dataset.
https://docs.arangodb.com/3.11/aql/examples-and-query-patterns/actors-and-movies-dataset-queries/, 2023. [Online; accessed 14-November-2023].
[2] ArangoDB. Comparing ArangoDB AQL to Neo4j https://arangodb.com/learn/graphs/comparing-arangodb-aql-neo4j-cypher/, 2023. accessed 2023].
[3] ArangoDB. NoSQL Performance Benchmark 2018 –
PostgreSQL, OrientDB, Neo4j and ArangoDB. https://arangodb.com/2018/02/ nosql-performance-benchmark-2018-mongodb-postgresql-orientdb-neo4j-arangodb/, 2023. [Online; accessed 2023].
[4] ArangoDB. Vision. https://arangodb.com/about-us/: :text=ArangoDB [Online; accessed 2023].
[5] P. Barceló, E. V. Kostylev, M. Monet, J. Pérez, J. L. Reutter, and J.-P. Silva. The expressive power of
graph neural networks as a query language. SIGMOD Rec., 49(2):6–17, dec 2020.
[6] DGL. dgl.data. https://docs.dgl.ai/api/python/dgl.data.html/, 2023. [Online; accessed 2023].
[7] M. Douze. Getting some data. https://github.com/facebookresearch/faiss/wiki/Getting-started/, 2020. [Online; posted 28-June-2020].
[8] F. Geerts. A query language perspective on graph learning. In Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS '23, page 373–379, New York, NY, USA, 2023. Association for Computing Machinery.
[9] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with GPUs. IEEE Transactions on Big Data, 7(3):535–547, 2019.
[10] Z. Liu, Y. Wang, J. Bernard, and T. Munzner. Visualizing graph neural networks with corgie: Corresponding a graph to its embedding. IEEE Transactions on Visualization and Computer Graphics, 28(6):2500–2516, 2022.
[11]W. L. Lju Lazarevic. Overview of the Neo4j Graph Data Platform. https://neo4j.com/developer-blog/overview-of-the-neo4j-graph-data-platform/, 2021. [Online; accessed 2023].
[12] Meta. Faiss: A library for efficient similarity search. https://engineering.fb.com/2017/03/29/ data-infrastructure/faiss-a-library-for-efficient-similarity-search/, 2017. [Online; posted 29-March-2017].
[13] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications, 2021.

Thank you!

Questions?