NODES 2023: GDS and GenAI

Daniel Bukowski, Graph Data Science CSA, Neo4j
Alex Gilmore, Associate Consulting Engineer, Neo4j

Introduction

About Us

Daniel Bukowski,
Graph Data Science
Customer Success Architect

Alexander Gilmore,
Associate Professional
Services Engineer

Appreciation

Many people have helped make this project and this presentation a reality:

- Ravi Ramanathan
- Ken Kane
- Dave Shiposh
- Kumar Subbiah Shunmugathai
- Adam Cowley
- Alison Cossette
- Alexander Fournier
- Michael Hunger
- Tomaz Bratanic

And to all of our other colleagues who tested Agent Neo and provided valuable feedback.

Session Overview

This is a session about using Neo4j Knowledge Graphs and Graph Data Science to analyze, understand, and improve the performance of grounded LLM applications.

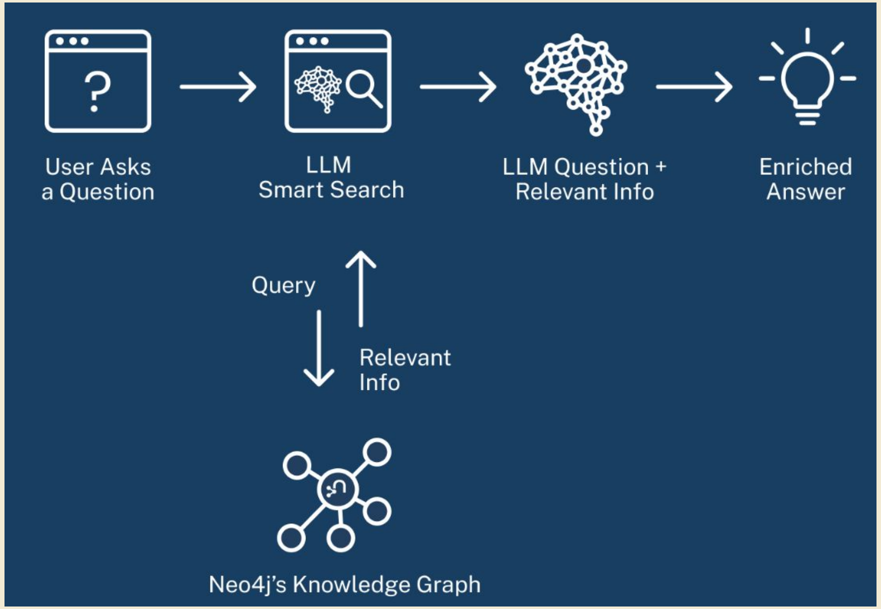
We will focus on:

- Building a high-quality grounding data set on a knowledge graph
- Logging LLM interactions as graphs
- Using graphs and Graph Data Science to improve the LLM's performance

We will briefly discuss how we built the LLM-powered assistant, but that is not the focus of the presentation.

Grounding and Retrieval Augmented Generation

- Grounding and RAG provides information to LLMs that they would not have available
- Grounding via knowledge graph is a primary use case for Neo4j with LLMs
- We found that grounding can significantly improve LLM responses about Neo4j and GDS
- RAG vs. Internet Search



Grounding and Fine-Tuning

- This presentation focuses on **grounding** because:
  - It is the primary method organizations will use to incorporate non-public data with LLM technology
  - Graphs and GDS are uniquely well suited to high-quality grounding and visualizing LLM performance

- **Fine-tuning** is a highly-effective way to further improve LLM performance by retraining part of the LLM specific questions and answers.
  - We find grounding is often the first step organizations take
  - Grounding is also an essential step to incorporate non-public information

Session Summary

- Grounding enables you to combine the power of LLMs with insights in your organization's internal data.

- High-quality data is essential for deploying a grounded LLM-based application into production.

- Neo4j and Graph Data Science are uniquely well suited to:
  - Develop, curate, and manage **high-quality grounding data at scale**
  - **Log LLM interactions** in the same database as the grounding data
  - Visualize LLM interactions with grounding data to **increase interpretability**
  - **Analyze LLM performance** to identify areas for improvement

LLM Application Overview

Why We Built This

- We started experimenting with ChatGPT earlier this year, but found it was hallucinating when asked questions about Graph Data Science

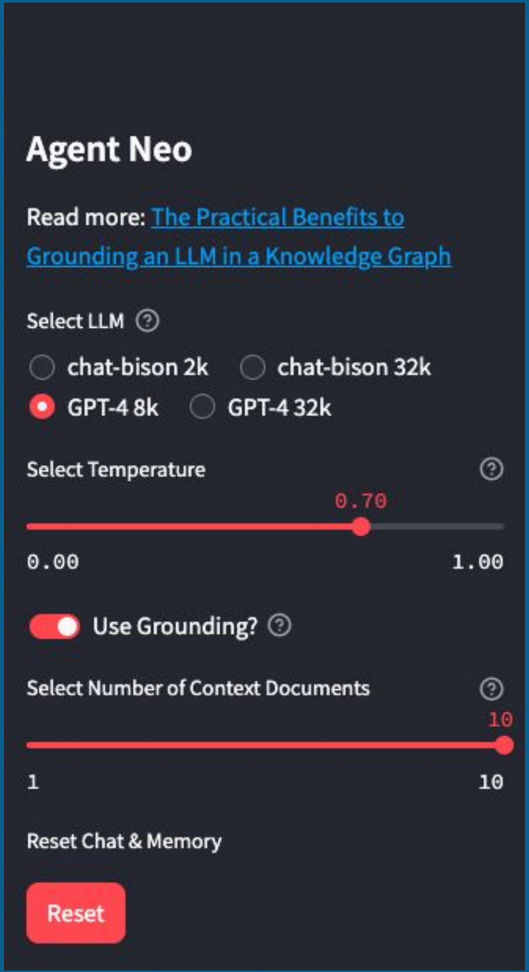
- Given prior September 2021 knowledge cutoffs, grounding was a way to provide the LLM updated information

- Our experimentation demonstrated ***significant improvements*** when the LLM was provided with grounding documents

- We moved the workflow from a Jupyter Notebook to a Streamlit front-end as the project gained momentum

Agent Neo Overview

- LLM Application to assist individuals with using the Neo4j Graph Data Science Library
- Grounded on a Neo4j AuraDS Knowledge Graph containing text from Neo4j Documentation
- Built using Streamlit and LangChain
- Logs conversations ***in the same graph database*** with context documents



Agent Neo

Read more: [The Practical Benefits to Grounding an LLM in a Knowledge Graph]

Select LLM

( ) chat-bison 2k ( ) chat-bison 32k
[✓ GPT-4 8k] ( ) GPT-4 32k

Select Temperature: 0.70 (0.00 – 1.00)

Use Grounding? [✓]

Select Number of Context Documents: 10 (1 – 10)

Reset Chat & Memory

Grounding Data Sources

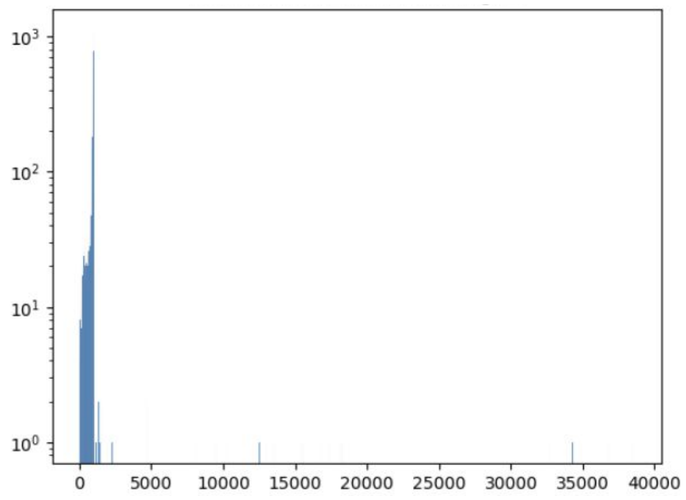
- Our grounding data set comprises 1,150 public Neo4j documents from official documentation, developer blogs, the support knowledge base, and Github

- We split this text into 15,000 embedded text ‘chunks’:
  - 512 chunk size because we were initially working with smaller context windows
  - LangChain Recursive Text Splitter
  - Embeddings via GCP

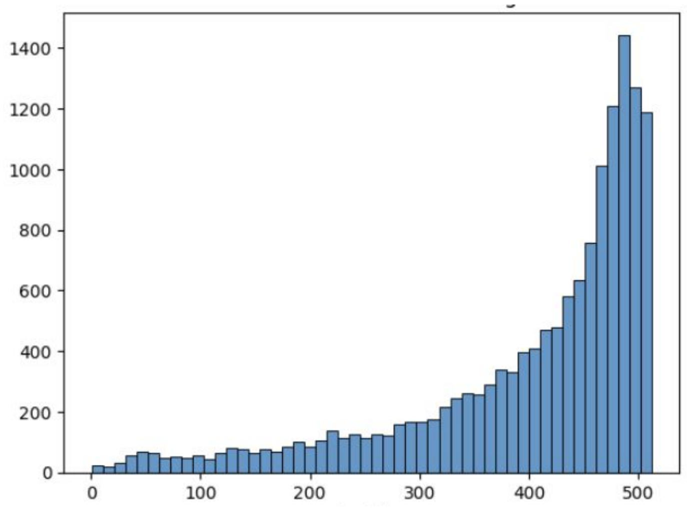
- We used public documents to have URLs associated with each text chunk and so the LLM could provide citations

EDA on Source Documents

EDA on source documents and document chunks is a critical step before generating embeddings and loading them into the database.

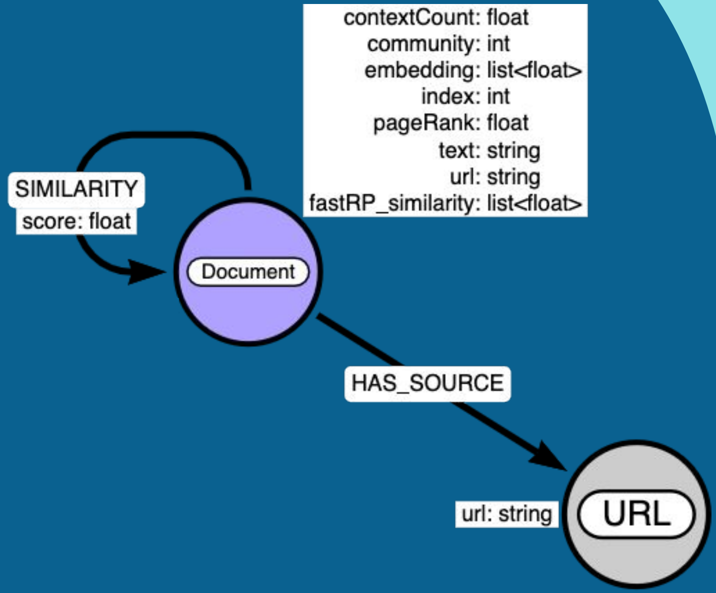


Text length distribution with chunking strategy errors



Text length distribution with corrected chunking strategy

Agent Neo Data Model



High-Quality Grounding via Graphs and GDS

LLM Grounding and Data-Centric AI

- Andrew Ng encourages “data-centric AI” which is an approach that focuses on creating and curating high-quality data sets to improve model performance

- Retrieval Augmented Generation (RAG) is Data-Centric AI
  - LLM trained on a massive, mixed-quality, data set
  - Improve responses by “grounding” them in a smaller, high-quality dataset

- Major differentiator for LLM performance will be the quality and efficiency of the grounding data set

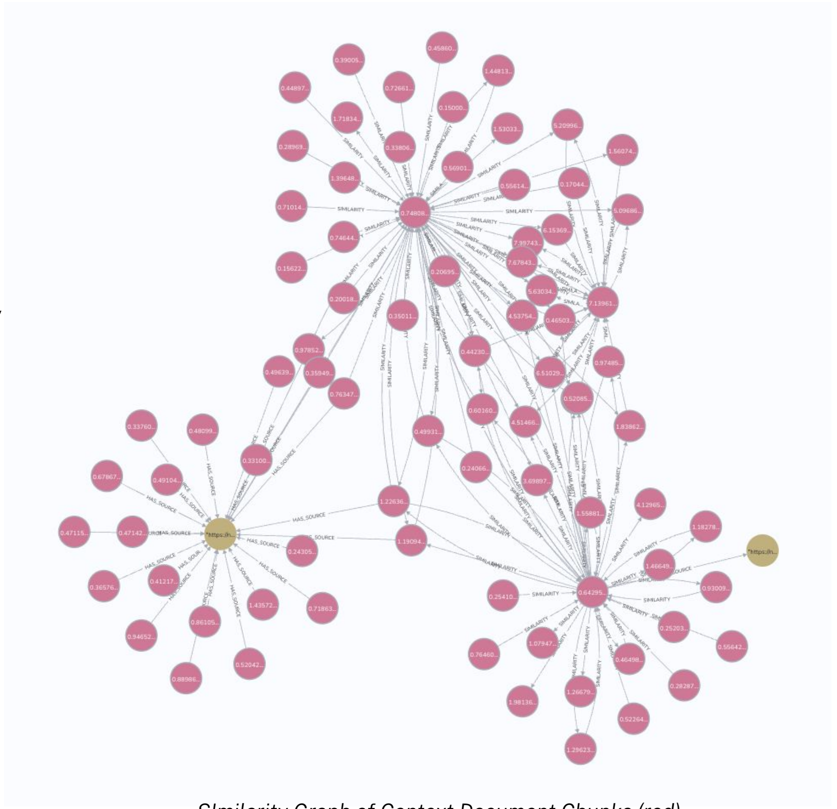
- Low-quality data can negatively impact grounded LLMs

Elements of High-Quality Grounding Data

Relevant	Related to the problem the LLM is solving and the questions you expect users to ask.
Augmenting	Fills known gaps in the LLM's 'knowledge', due to data being non-public our outside the training window.
Reliable	Contains accurate information, whether from inside or outside of the organization.
Clean	Is generally free of errors or noise, especially if generated from notebooks, websites, repos, etc...
Efficient	Does not contain duplicates, or near-duplicate, 'chunks' that take up valuable context limits.

EDA with Graphs and GDS

- Traditional preprocessing and EDA on the source text is critical
- Use GDS to create **KNN Similarity** relationships among context nodes
- Gain additional insights via **Community Detection** and **Centrality**
- Capture the similarity graph via **Node Embeddings**
- Curate the grounding data set via techniques that work **at scale**

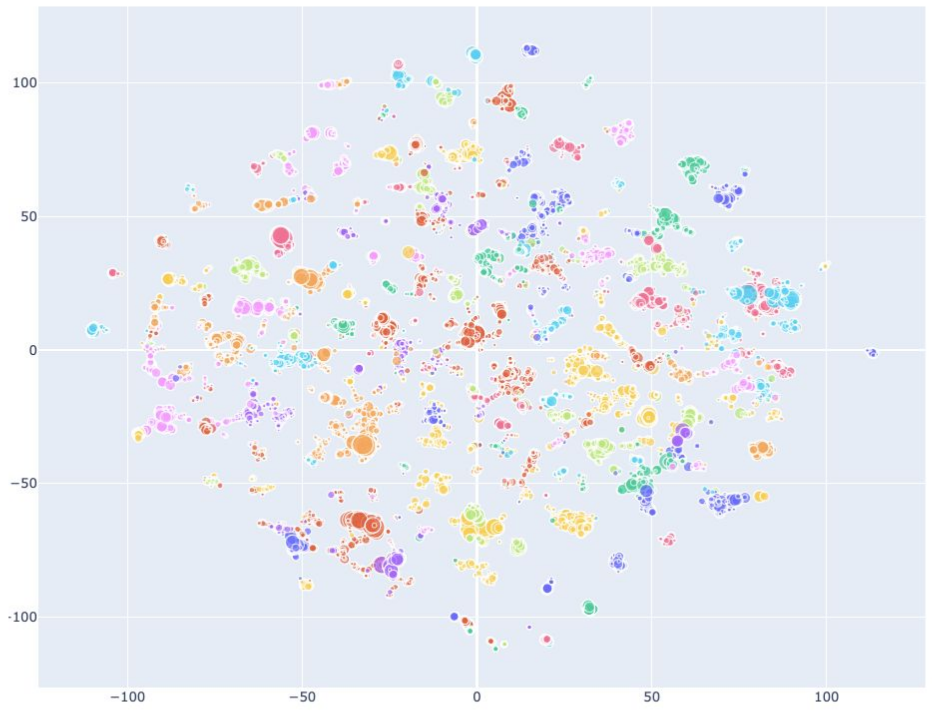

SImilarity Graph of Context Document Chunks (red) with Source URLs (gold)

Clean

Analyzing Context with Graphs and GDS

The visualization shows:

- FastRP embeddings of context documents
- Icon color represents the LPA graph community
- Icon size represents the context chunk's PageRank

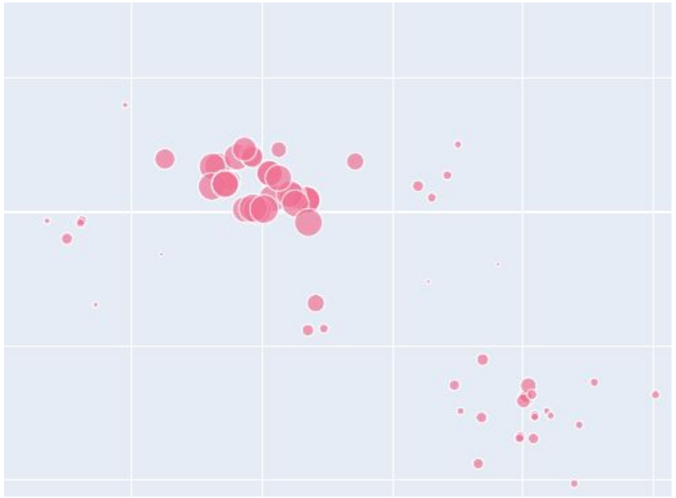


2D Visualization of Context Document Node Embeddings

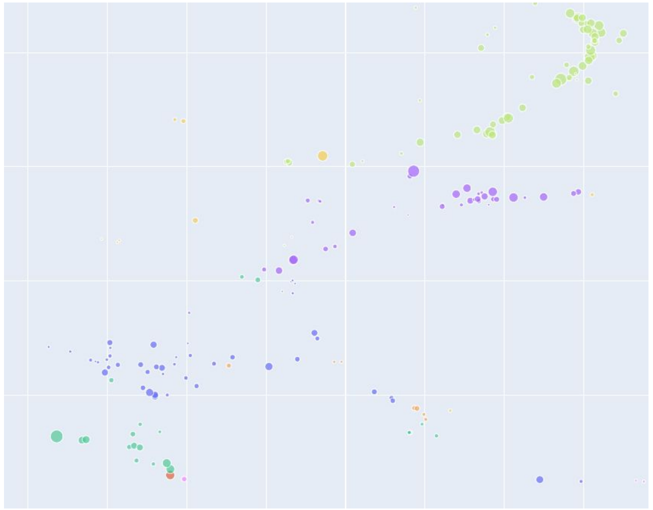
Embedding Visualization Detail

Clean

The document embeddings are generally distinct, but some communities overlap.



Single-Community Cluster

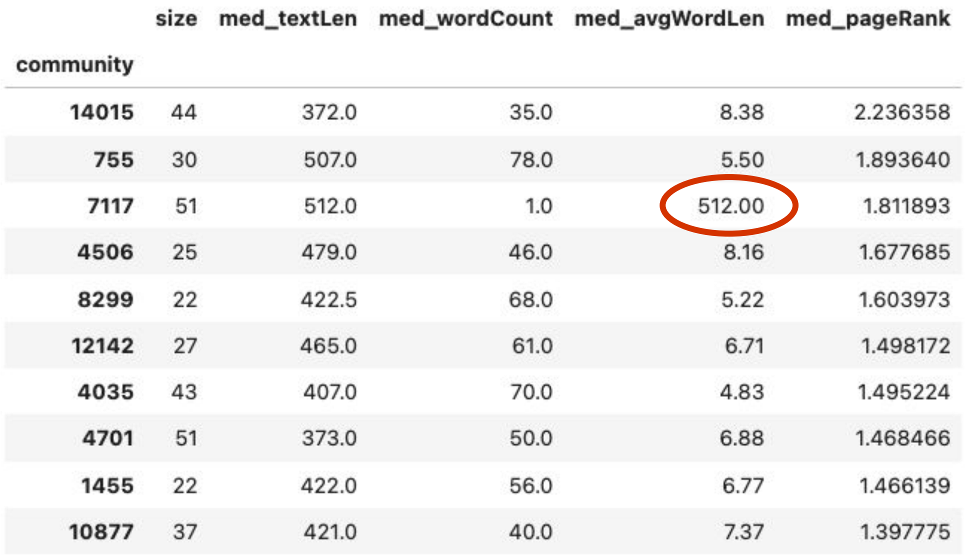


Overlapping Communities

Identifying Text Errors

Combining graph and traditional statistics helps us identify outliers or data quality issues.

- Traditional: Text length, word count, and word length
- Graph: Communities, community size, and PageRank score

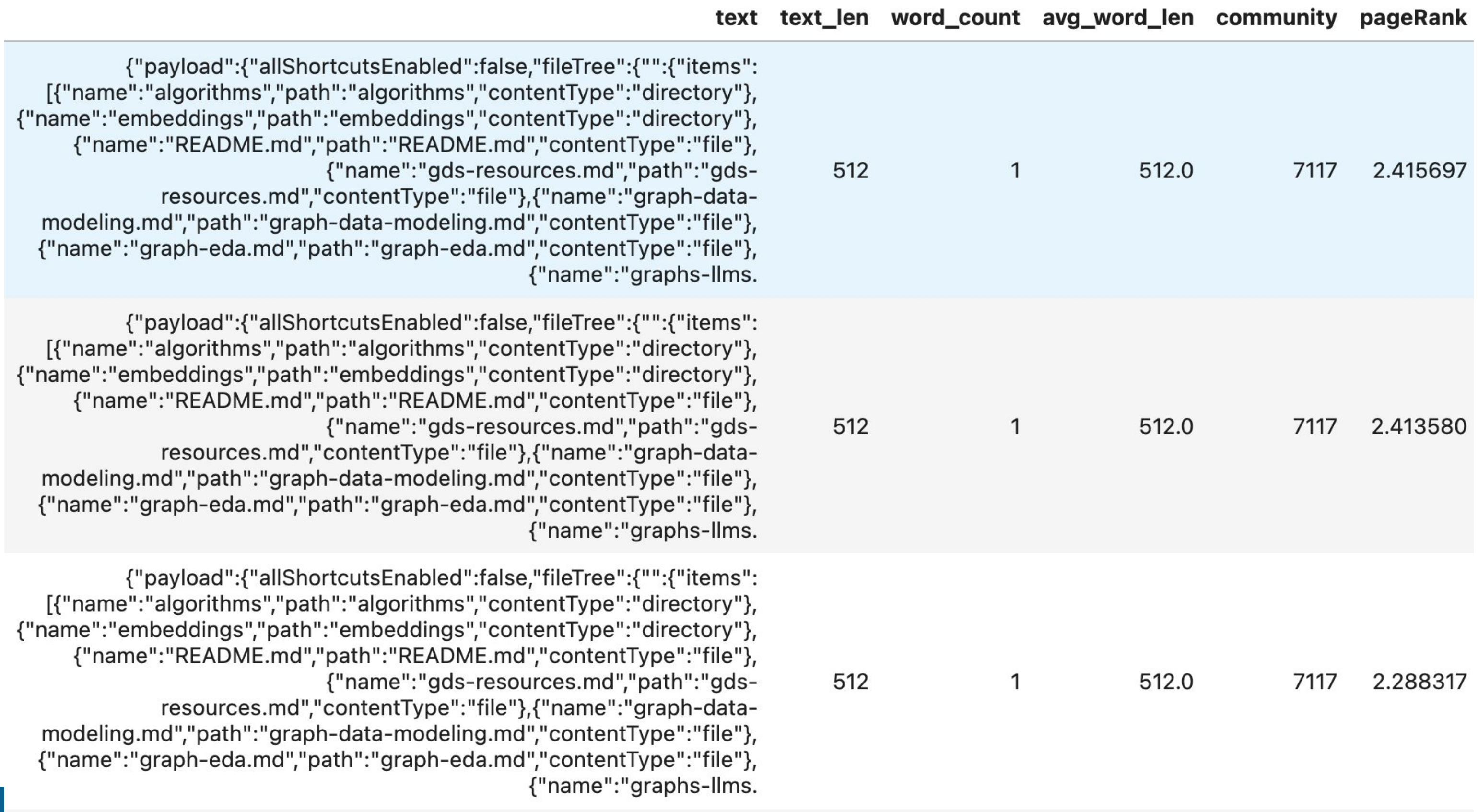


	size	med_textLen	med_wordCount	med_avgWordLen	med_pageRank
community					
14015	44	372.0	35.0	8.38	2.236358
755	30	507.0	78.0	5.50	1.893640
7117	51	512.0	1.0	512.00	1.811893
4506	25	479.0	46.0	8.16	1.677685
8299	22	422.5	68.0	5.22	1.603973
12142	27	465.0	61.0	6.71	1.498172
4035	43	407.0	70.0	4.83	1.495224
4701	51	373.0	50.0	6.88	1.468466
1455	22	422.0	56.0	6.77	1.466139
10877	37	421.0	40.0	7.37	1.397775

Graph Communities with additional statistics;
Note: Outlier average word length in community 7117

Clean

Investigating Text Chunk Outliers



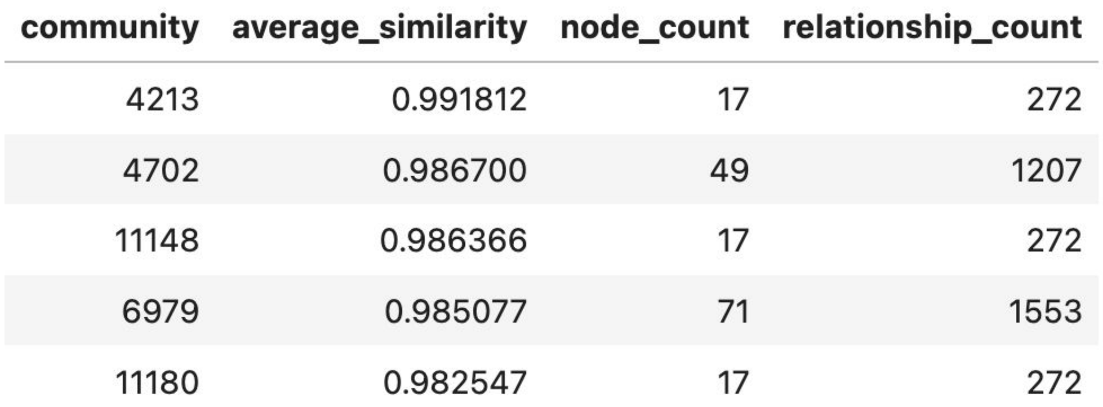
text	text_len	word_count	avg_word_len	community	pageRank
{"payload":{"allShortcutsEnabled":false,"fileTree":{"":{"items":[{"name":"algorithms","path":"algorithms","contentType":"directory"},{"name":"embeddings","path":"embeddings","contentType":"directory"},{"name":"README.md","path":"README.md","contentType":"file"},{"name":"gds-resources.md","path":"gds-resources.md","contentType":"file"},{"name":"graph-data-modeling.md","path":"graph-data-modeling.md","contentType":"file"},{"name":"graph-eda.md","path":"graph-eda.md","contentType":"file"},{"name":"graphs-llms.	512	1	512.0	7117	2.415697
{"payload":{"allShortcutsEnabled":false,"fileTree":{"":{"items":[{"name":"algorithms","path":"algorithms","contentType":"directory"},{"name":"embeddings","path":"embeddings","contentType":"directory"},{"name":"README.md","path":"README.md","contentType":"file"},{"name":"gds-resources.md","path":"gds-resources.md","contentType":"file"},{"name":"graph-data-modeling.md","path":"graph-data-modeling.md","contentType":"file"},{"name":"graph-eda.md","path":"graph-eda.md","contentType":"file"},{"name":"graphs-llms.	512	1	512.0	7117	2.413580
{"payload":{"allShortcutsEnabled":false,"fileTree":{"":{"items":[{"name":"algorithms","path":"algorithms","contentType":"directory"},{"name":"embeddings","path":"embeddings","contentType":"directory"},{"name":"README.md","path":"README.md","contentType":"file"},{"name":"gds-resources.md","path":"gds-resources.md","contentType":"file"},{"name":"graph-data-modeling.md","path":"graph-data-modeling.md","contentType":"file"},{"name":"graph-eda.md","path":"graph-eda.md","contentType":"file"},{"name":"graphs-llms.	512	1	512.0	7117	2.288317

Efficient

Highly-Similar Text Chunks

Because we persisted the KNN Similarity relationships, we can query them just like any other object in the graph:

- Count 99%+ similarity relationships in the database?
- Identify all text chunks with at least one 99%+ relationship
- Identify communities with the highest average similarity scores



community	average_similarity	node_count	relationship_count
4213	0.991812	17	272
4702	0.986700	49	1207
11148	0.986366	17	272
6979	0.985077	71	1553
11180	0.982547	17	272

Efficient

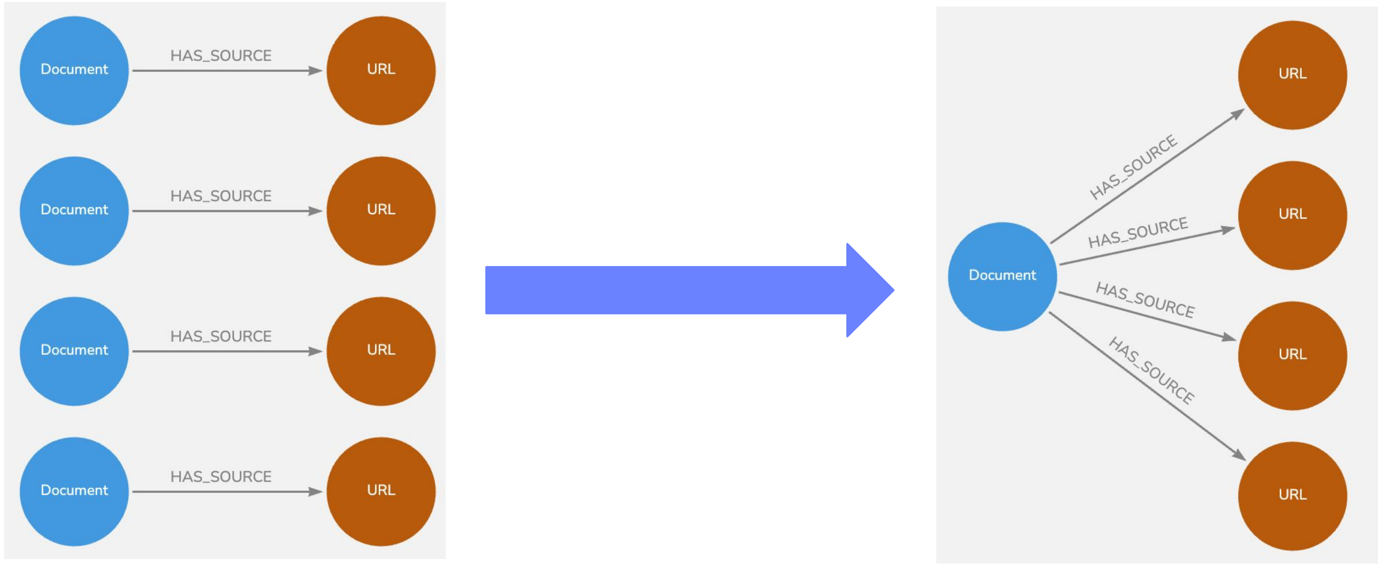
Highly-Similar Text Chunks

url	text	community
https://neo4j.com/docs/graph-data-science/current/machine-learning/node-embeddings/graph-sage/	Heterogeneous relationships Heterogeneous relationships fully supported. The algorithm has the ability to distinguish between relationships of different types. Heterogeneous relationships Heterogeneous relationships allowed. The algorithm treats all selected relationships similarly regardless of their type. Weighted relationships Weighted trait. The algorithm supports a relationship property to be used as weight, specified via the relationshipWeightProperty configuration parameter.	4702
https://neo4j.com/docs/graph-data-science/current/algorithms/harmonic-centrality/	Heterogeneous relationships Heterogeneous relationships fully supported. The algorithm has the ability to distinguish between relationships of different types. Heterogeneous relationships Heterogeneous relationships allowed. The algorithm treats all selected relationships similarly regardless of their type. Weighted relationships Weighted trait. The algorithm supports a relationship property to be used as weight, specified via the relationshipWeightProperty configuration parameter.	4702
https://neo4j.com/docs/graph-data-science/current/algorithms/bellman-ford-single-source/	Heterogeneous relationships Heterogeneous relationships fully supported. The algorithm has the ability to distinguish between relationships of different types. Heterogeneous relationships Heterogeneous relationships allowed. The algorithm treats all selected relationships similarly regardless of their type. Weighted relationships Weighted trait. The algorithm supports a relationship property to be used as weight, specified via the relationshipWeightProperty configuration parameter.	4702
https://neo4j.com/docs/graph-data-science/current/algorithms/modularity-optimization/	Heterogeneous relationships Heterogeneous relationships fully supported. The algorithm has the ability to distinguish between relationships of different types. Heterogeneous relationships Heterogeneous relationships allowed. The algorithm treats all selected relationships similarly regardless of their type. Weighted relationships Weighted trait. The algorithm supports a relationship property to be used as weight, specified via the relationshipWeightProperty configuration parameter.	4702
https://neo4j.com/docs/graph-data-science/current/algorithms/closeness-centrality/	Heterogeneous relationships Heterogeneous relationships fully supported. The algorithm has the ability to distinguish between relationships of different types. Heterogeneous relationships Heterogeneous relationships allowed. The algorithm treats all selected relationships similarly regardless of their type. Weighted relationships Weighted trait. The algorithm supports a relationship property to be used as weight, specified via the relationshipWeightProperty configuration parameter.	4702

Efficient

Collapsing Duplicate Nodes

We can use `apoc.nodes.collapse()` to combine duplicate nodes into a single node with all prior relationships pointing to the single node.



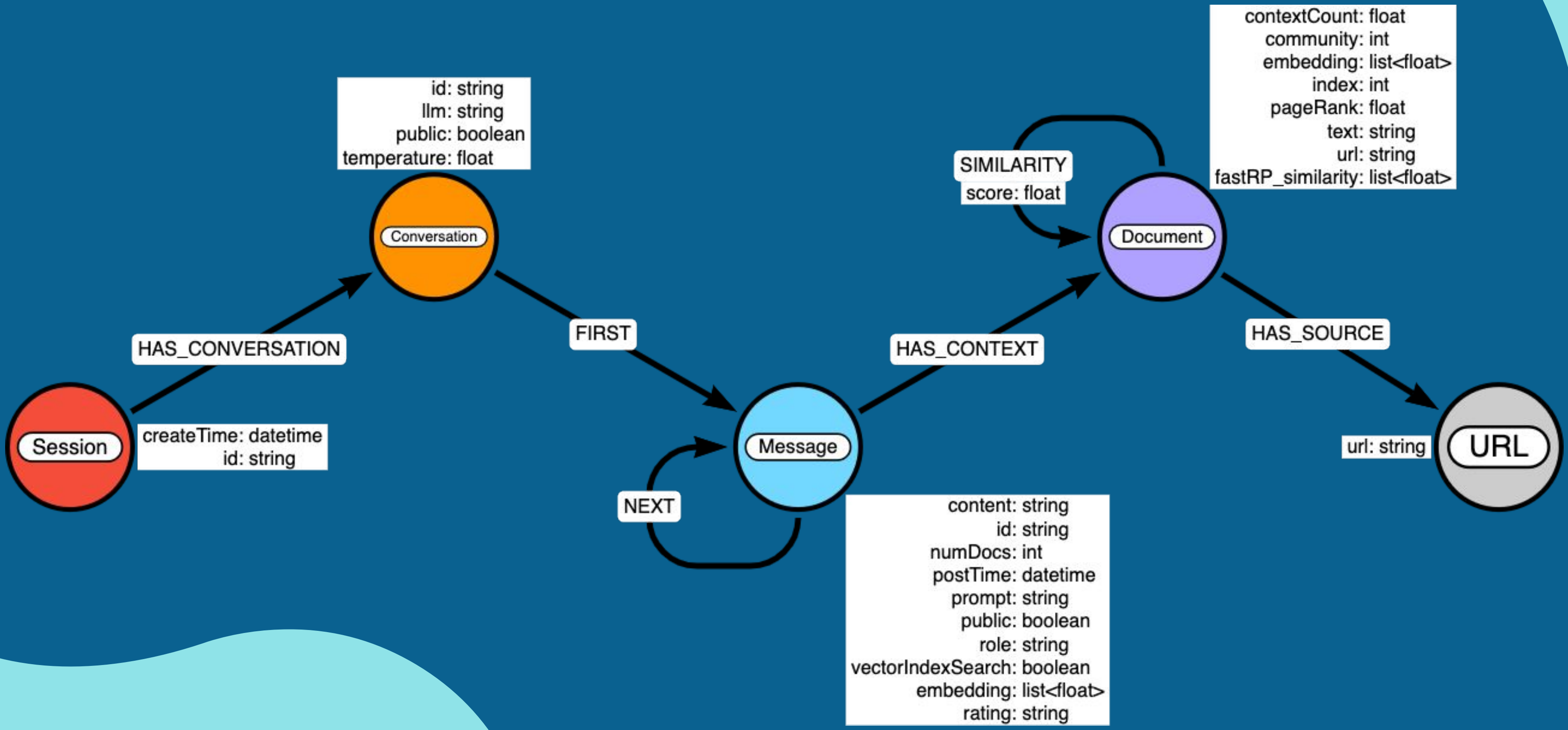
Logging Conversations with the LLM

Graphs Enable Explainable AI with LLMs

- As LLM applications enter production within organizations, understanding how they use grounding documents and how they produce answers will become more and more important

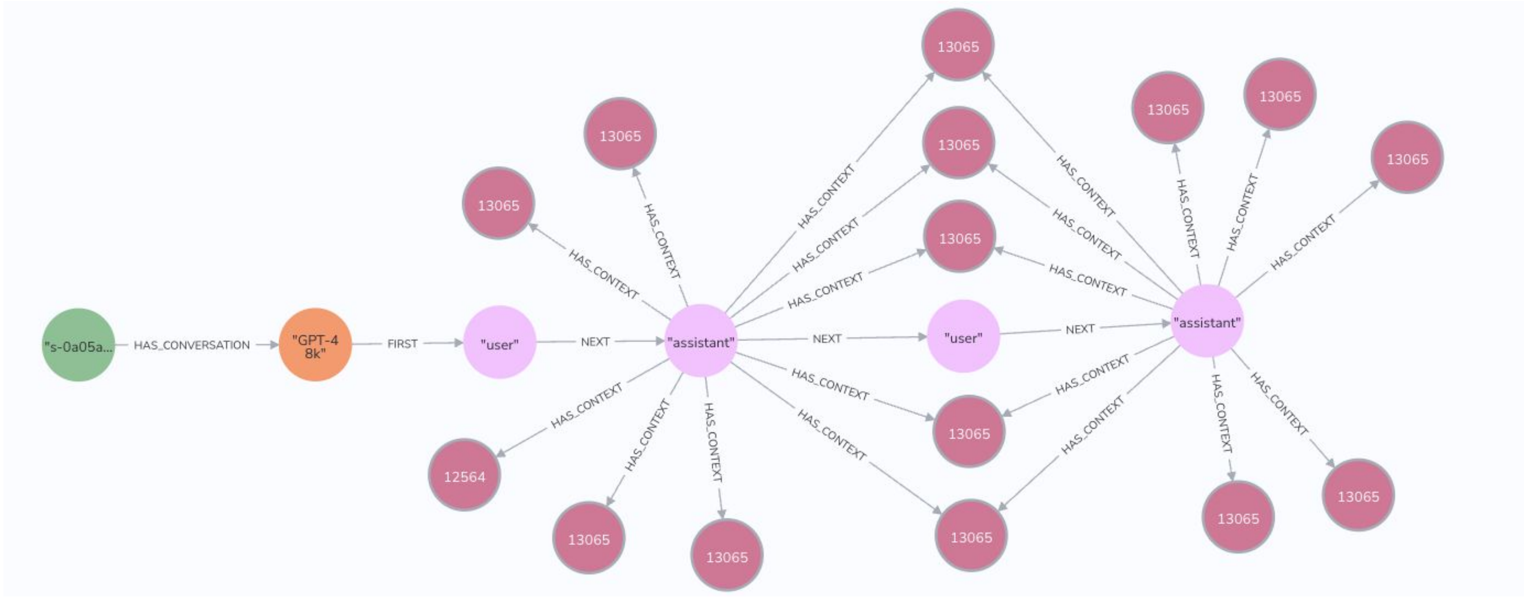
- Knowledge Graphs and GDS enable Explainable AI by:
  - Logging user interactions in the same database as the context
  - Visualizing conversations with context
  - Providing tools to analyze LLM performance and identify opportunities for improvement

Full Agent Neo Data Model



Logging and Visualizing Conversations

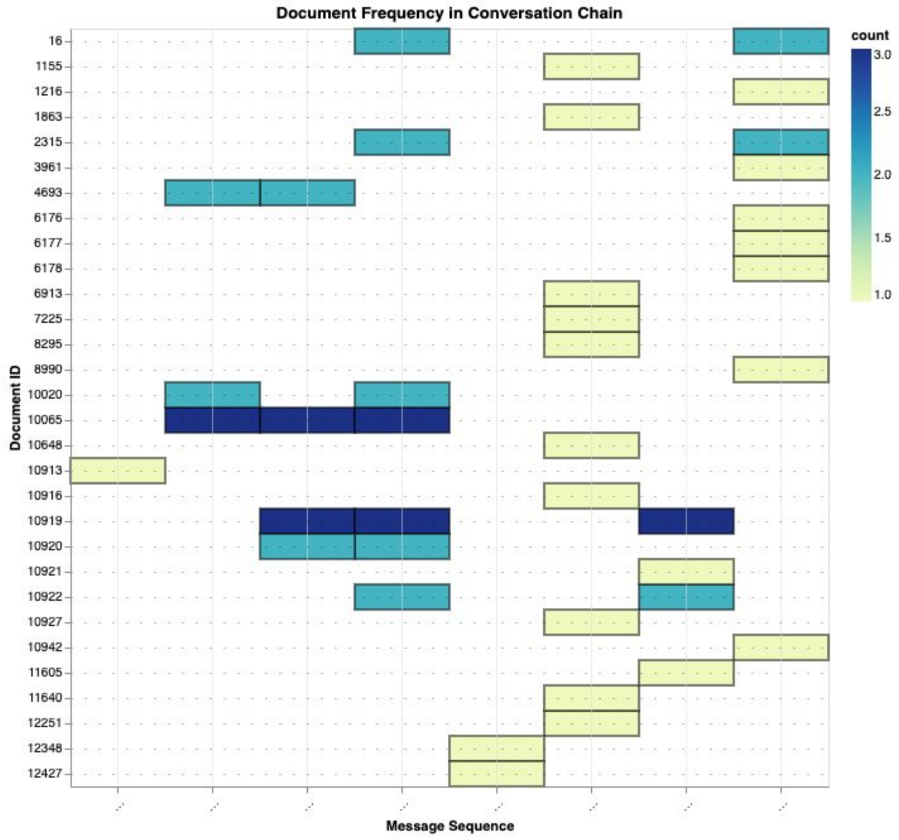
Graphs enable logging of LLM conversations in the same database as the context documents and with defined relationships.



Graph of an actual conversation between an Agent Neo user and the ChatGPT-4 LLM. Context Documents are labeled with their GDS Community.

Visualizing Document Usage

- The heat map depicts how frequently documents are used during a single conversation:
  - X-axis represents LLM messages
  - Y-axis represents individual documents
  - Color depicts document use frequency count (1x to 3x)
- Documents are re-used throughout conversations



Analyzing LLM Conversations

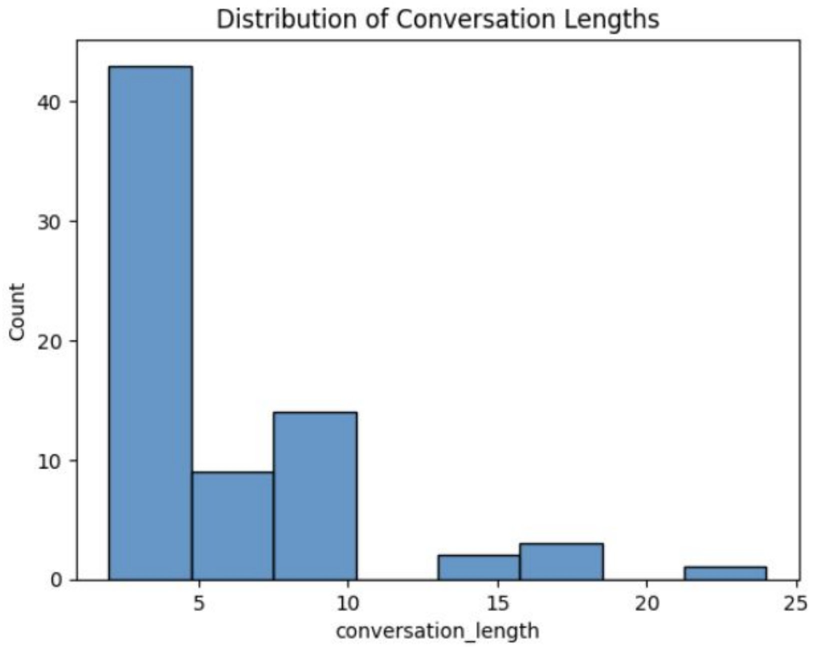
Use NeoDash to Monitor LLM Metrics

Relevant, Augmenting, Reliable

- NeoDash is an open-source, low-code, community-supported Dashboard Builder for Neo4j.
- Designed for Neo4j, NeoDash enables you to build an interactive dashboard with tables, graphs, bar charts, line charts, maps and more.
- A NeoDash dashboard can easily visualize key metrics about your grounded LLM application, including about the grounding documentation and conversations.
- For more information about NeoDash:
    - Road to NODES 2023 Workshop **Intro to Neo4j & Interactive Dashboarding with NeoDash** available on Neo4j's YouTube page
    - https://neo4j.com/labs/neodash/

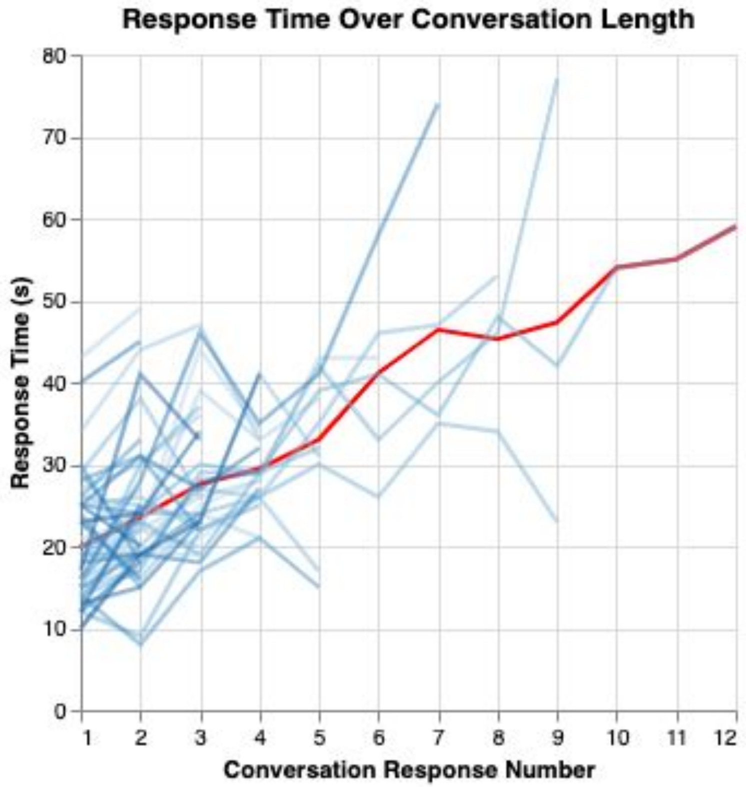
Conversation Lengths

- Because conversations are logged as graphs it is easy to measure the length and store it as a new property on each Conversation node.
- Most users have been experimenting with our tool, so we expect conversation lengths to be shorter.
- As users are more comfortable with these applications, we expect conversation lengths to increase.



LLM Response Time

- Our logging identified that as the conversation length increases, the response time also increases.
- Other factors to consider with response time include:
  - LLM rate limits
  - Model context windows
  - Conversation memory



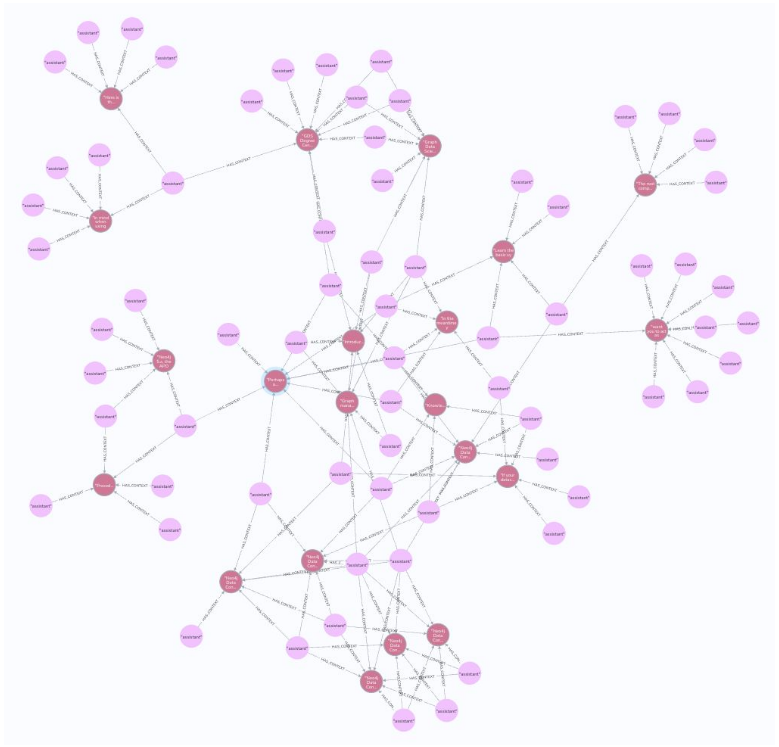
Graph-Based Analysis of Context Document Usage

Relevant

Visualizing Context Document Usage

- Graphs enable us to visualize the most frequently used context Documents along with the associated LLM responses

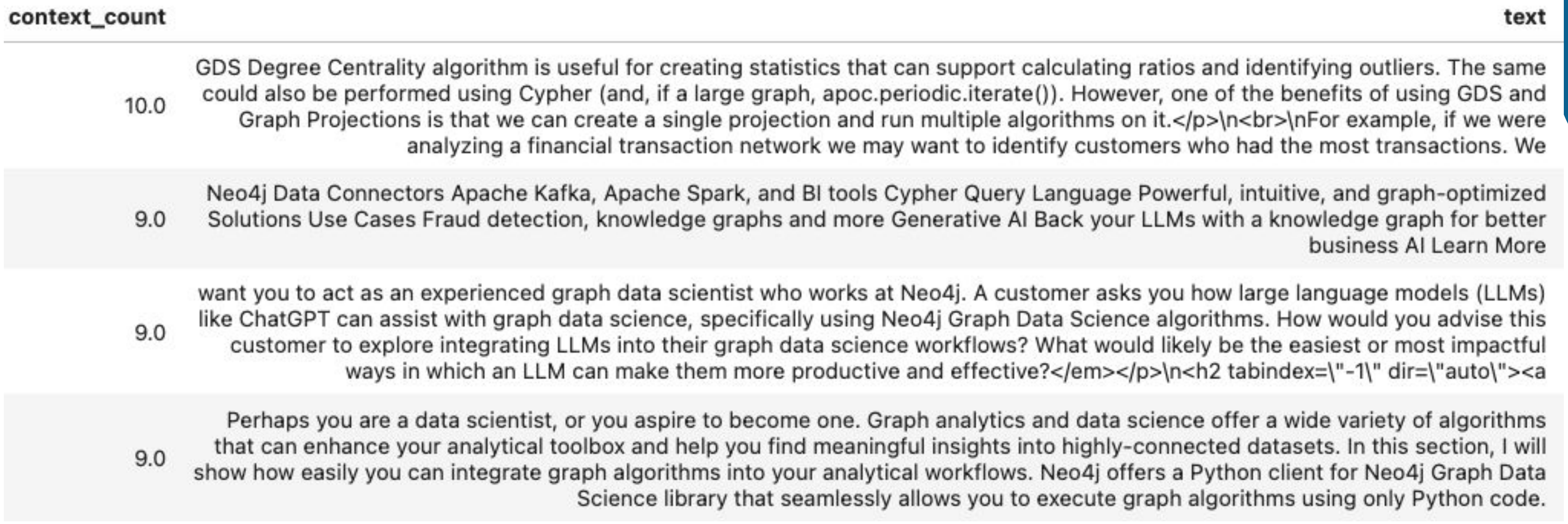
- Natural clusters form in the graph even among the most frequently used Documents



LLM Responses (pink) and
Most Frequently Used Context Documents (red)

Most Frequently Used Grounding Text

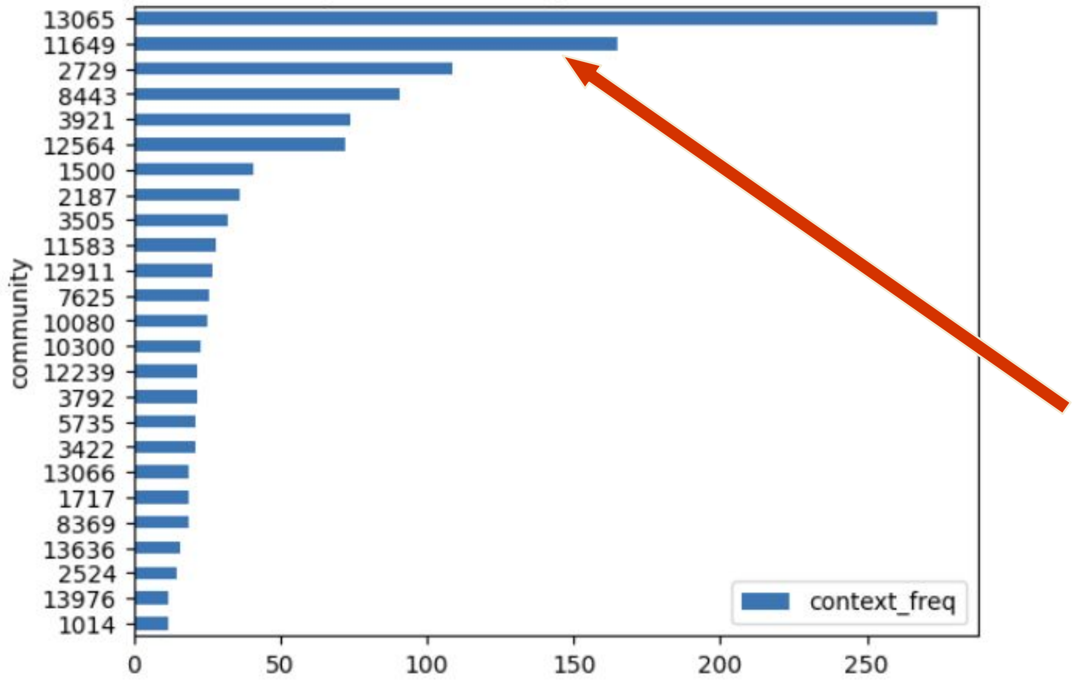
Relevant



context_count	text
10.0	GDS Degree Centrality algorithm is useful for creating statistics that can support calculating ratios and identifying outliers. The same could also be performed using Cypher (and, if a large graph, apoc.periodic.iterate()). However, one of the benefits of using GDS and Graph Projections is that we can create a single projection and run multiple algorithms on it.</p>\n \nFor example, if we were analyzing a financial transaction network we may want to identify customers who had the most transactions. We
9.0	Neo4j Data Connectors Apache Kafka, Apache Spark, and BI tools Cypher Query Language Powerful, intuitive, and graph-optimized Solutions Use Cases Fraud detection, knowledge graphs and more Generative AI Back your LLMs with a knowledge graph for better business AI Learn More
9.0	want you to act as an experienced graph data scientist who works at Neo4j. A customer asks you how large language models (LLMs) like ChatGPT can assist with graph data science, specifically using Neo4j Graph Data Science algorithms. How would you advise this customer to explore integrating LLMs into their graph data science workflows? What would likely be the easiest or most impactful ways in which an LLM can make them more productive and effective?</em></p>\n<h2 tabindex=\"-1\" dir=\"auto\"><a
9.0	Perhaps you are a data scientist, or you aspire to become one. Graph analytics and data science offer a wide variety of algorithms that can enhance your analytical toolbox and help you find meaningful insights into highly-connected datasets. In this section, I will show how easily you can integrate graph algorithms into your analytical workflows. Neo4j offers a Python client for Neo4j Graph Data Science library that seamlessly allows you to execute graph algorithms using only Python code.

Relevant

Most Frequent Document Communities



- Combining Document usage frequency with previously identified Document Communities, we can see which of these Communities are the most frequent sources of Context Documents

- The second most frequent Community (11649) comprises text chunks from blogs by Tomaz Bratanic

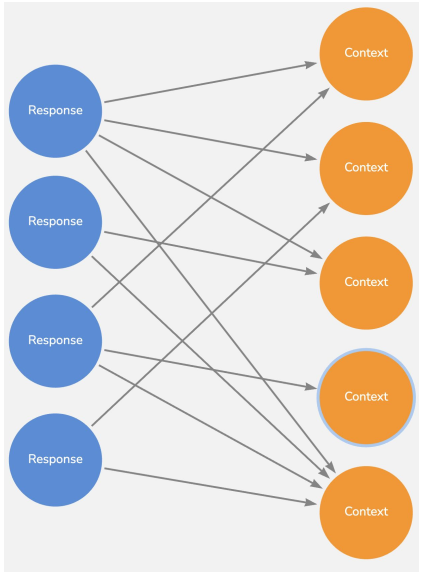
Analyzing LLM Responses Based Upon Context Documents

Analyzing LLM Responses by Context Docs

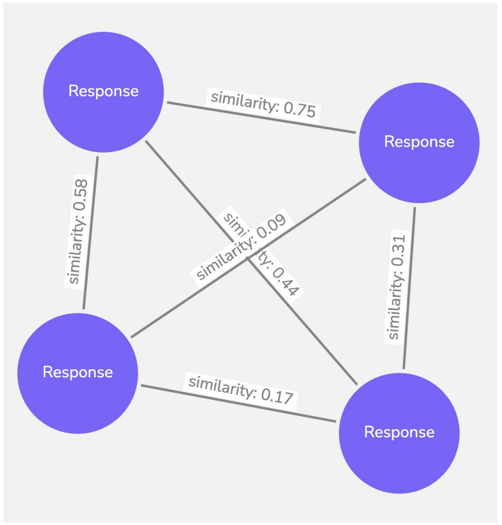
We can use GDS Algorithms to analyze the LLM Responses based upon the Context Documents used to generate those responses.

First we project a bi-partite graph of LLM Response nodes and Context Document nodes.

Then we apply the GDS Node Similarity algorithm to create a similarity graph of the LLM Response nodes.



Bi-Partite Graph



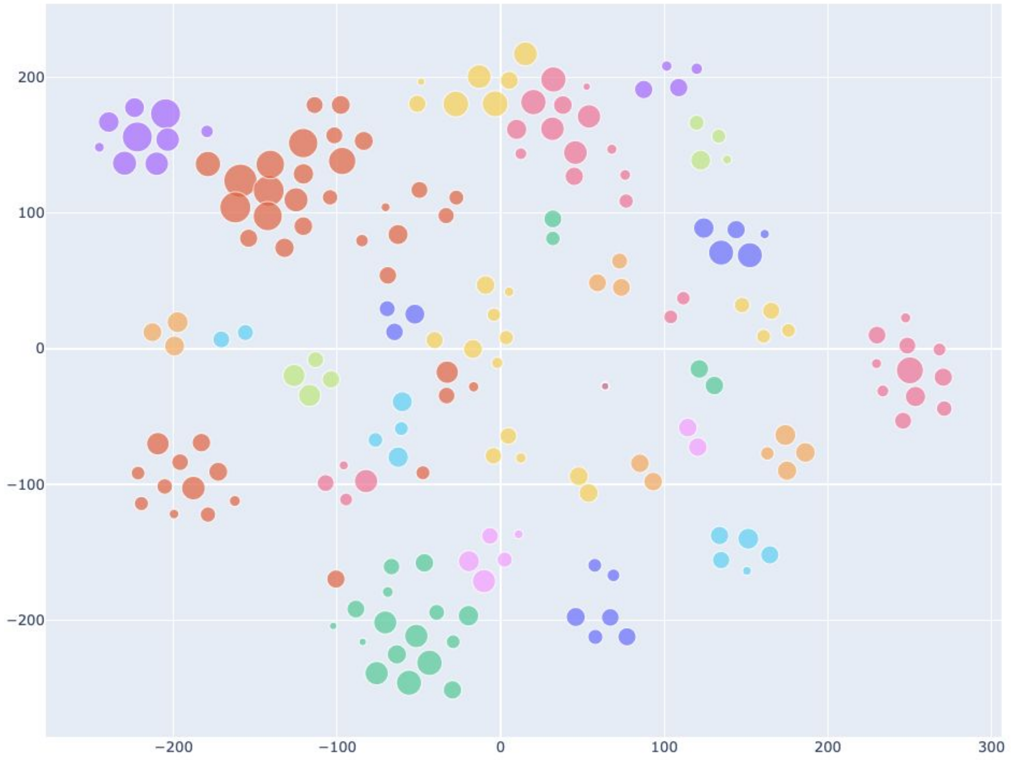
Similarity Graph

Relevant, Clean, Reliable

Visualize LLM Response Embeddings

The visualization shows:

- Icon color represents the graph community
- Icon size represents the response node's PageRank
- FastRP embedding of LLM Response similarity graph

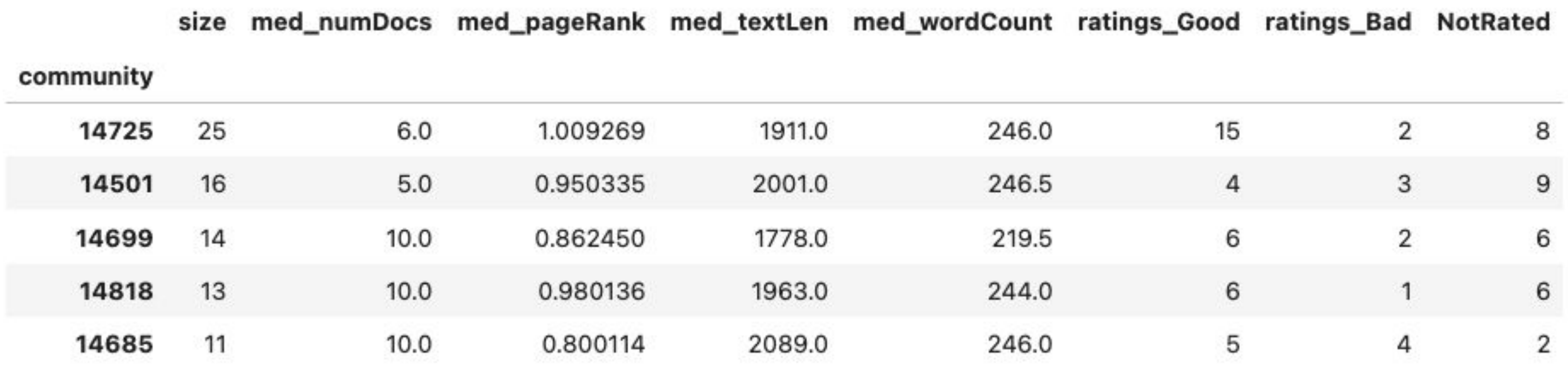


2D Plot of LLM Response Similarity Graph

Relevant, Augmenting, Clean

Inspect Communities of LLM Responses

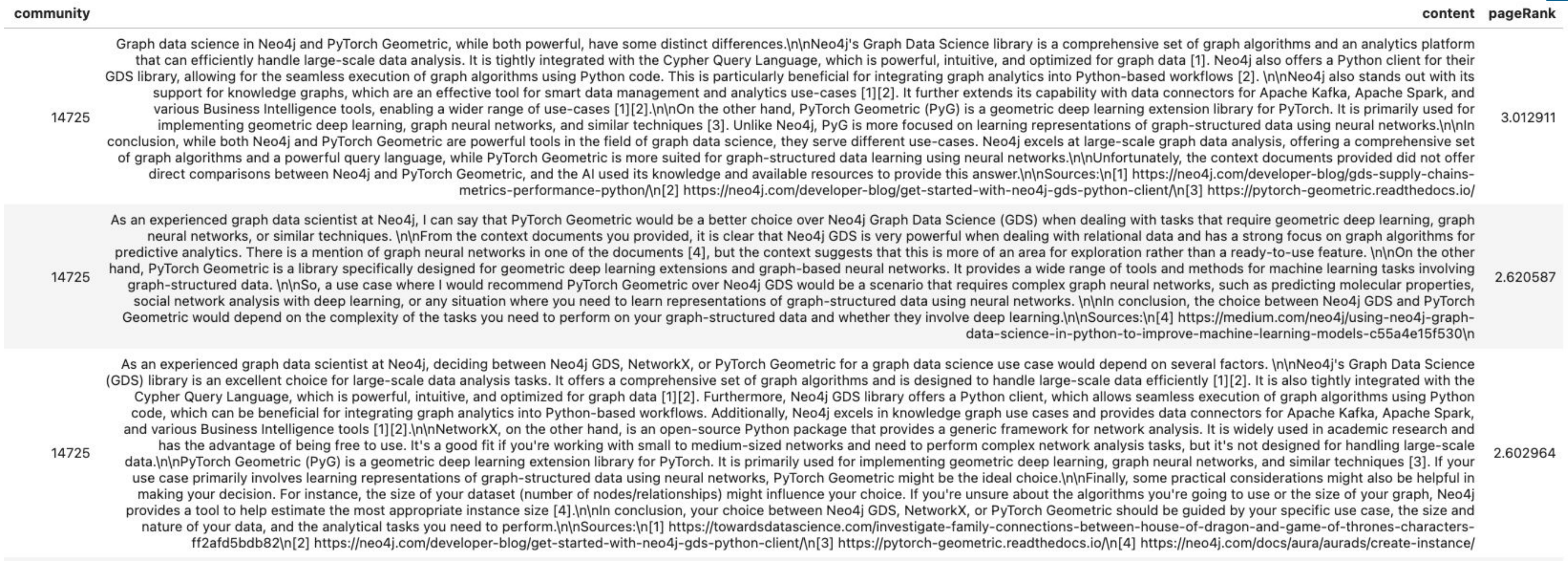
We can use similar approaches to inspect each of the LLM response communities via traditional and GDS-based statistics.



community	size	med_numDocs	med_pageRank	med_textLen	med_wordCount	ratings_Good	ratings_Bad	NotRated
14725	25	6.0	1.009269	1911.0	246.0	15	2	8
14501	16	5.0	0.950335	2001.0	246.5	4	3	9
14699	14	10.0	0.862450	1778.0	219.5	6	2	6
14818	13	10.0	0.980136	1963.0	244.0	6	1	6
14685	11	10.0	0.800114	2089.0	246.0	5	4	2

LLM Responses with Highest PageRank Scores

Note: The responses below all mention GDS, Network X, and PyTorch Geometric



community	content	pageRank
14725	Graph data science in Neo4j and PyTorch Geometric, while both powerful, have some distinct differences.\n\nNeo4j's Graph Data Science library is a comprehensive set of graph algorithms and an analytics platform that can efficiently handle large-scale data analysis. It is tightly integrated with the Cypher Query Language, which is powerful, intuitive, and optimized for graph data [1]. Neo4j also offers a Python client for their GDS library, allowing for the seamless execution of graph algorithms using Python code. This is particularly beneficial for integrating graph analytics into Python-based workflows [2]. \n\nNeo4j also stands out with its support for knowledge graphs, which are an effective tool for smart data management and analytics use-cases [1][2]. It further extends its capability with data connectors for Apache Kafka, Apache Spark, and various Business Intelligence tools, enabling a wider range of use-cases [1][2].\n\nOn the other hand, PyTorch Geometric (PyG) is a geometric deep learning extension library for PyTorch. It is primarily used for implementing geometric deep learning, graph neural networks, and similar techniques [3]. Unlike Neo4j, PyG is more focused on learning representations of graph-structured data using neural networks.\n\nIn conclusion, while both Neo4j and PyTorch Geometric are powerful tools in the field of graph data science, they serve different use-cases. Neo4j excels at large-scale graph data analysis, offering a comprehensive set of graph algorithms and a powerful query language, while PyTorch Geometric is more suited for graph-structured data learning using neural networks.\n\nUnfortunately, the context documents provided did not offer direct comparisons between Neo4j and PyTorch Geometric, and the AI used its knowledge and available resources to provide this answer.\n\nSources:\n[1] https://neo4j.com/developer-blog/gds-supply-chains-metrics-performance-python/\n[2] https://neo4j.com/developer-blog/get-started-with-neo4j-gds-python-client/\n[3] https://pytorch-geometric.readthedocs.io/	3.012911
14725	As an experienced graph data scientist at Neo4j, I can say that PyTorch Geometric would be a better choice over Neo4j Graph Data Science (GDS) when dealing with tasks that require geometric deep learning, graph neural networks, or similar techniques. \n\nFrom the context documents you provided, it is clear that Neo4j GDS is very powerful when dealing with relational data and has a strong focus on graph algorithms for predictive analytics. There is a mention of graph neural networks in one of the documents [4], but the context suggests that this is more of an area for exploration rather than a ready-to-use feature. \n\nOn the other hand, PyTorch Geometric is a library specifically designed for geometric deep learning extensions and graph-based neural networks. It provides a wide range of tools and methods for machine learning tasks involving graph-structured data. \n\nSo, a use case where I would recommend PyTorch Geometric over Neo4j GDS would be a scenario that requires complex graph neural networks, such as predicting molecular properties, social network analysis with deep learning, or any situation where you need to learn representations of graph-structured data using neural networks. \n\nIn conclusion, the choice between Neo4j GDS and PyTorch Geometric would depend on the complexity of the tasks you need to perform on your graph-structured data and whether they involve deep learning.\n\nSources:\n[4] https://medium.com/neo4j/using-neo4j-graph-data-science-in-python-to-improve-machine-learning-models-c55a4e15f530\n	2.620587
14725	As an experienced graph data scientist at Neo4j, deciding between Neo4j GDS, NetworkX, or PyTorch Geometric for a graph data science use case would depend on several factors. \n\nNeo4j's Graph Data Science (GDS) library is an excellent choice for large-scale data analysis tasks. It offers a comprehensive set of graph algorithms and is designed to handle large-scale data efficiently [1][2]. It is also tightly integrated with the Cypher Query Language, which is powerful, intuitive, and optimized for graph data [1][2]. Furthermore, Neo4j GDS library offers a Python client, which allows seamless execution of graph algorithms using Python code, which can be beneficial for integrating graph analytics into Python-based workflows. Additionally, Neo4j excels in knowledge graph use cases and provides data connectors for Apache Kafka, Apache Spark, and various Business Intelligence tools [1][2].\n\nNetworkX, on the other hand, is an open-source Python package that provides a generic framework for network analysis. It is widely used in academic research and has the advantage of being free to use. It's a good fit if you're working with small to medium-sized networks and need to perform complex network analysis tasks, but it's not designed for handling large-scale data.\n\nPyTorch Geometric (PyG) is a geometric deep learning extension library for PyTorch. It is primarily used for implementing geometric deep learning, graph neural networks, and similar techniques [3]. If your use case primarily involves learning representations of graph-structured data using neural networks, PyTorch Geometric might be the ideal choice.\n\nFinally, some practical considerations might also be helpful in making your decision. For instance, the size of your dataset (number of nodes/relationships) might influence your choice. If you're unsure about the algorithms you're going to use or the size of your graph, Neo4j provides a tool to help estimate the most appropriate instance size [4].\n\nIn conclusion, your choice between Neo4j GDS, NetworkX, or PyTorch Geometric should be guided by your specific use case, the size and nature of your data, and the analytical tasks you need to perform.\n\nSources:\n[1] https://towardsdatascience.com/investigate-family-connections-between-house-of-dragon-and-game-of-thrones-characters-ff2afd5bdb82\n[2] https://neo4j.com/developer-blog/get-started-with-neo4j-gds-python-client/\n[3] https://pytorch-geometric.readthedocs.io/\n[4] https://neo4j.com/docs/aura/aurads/create-instance/	2.602964

Conclusion

Summary

- Grounding enables you to combine the power of LLMs with insights in your organization's internal data.

- High-quality data is essential for deploying a grounded LLM-based application into production.

- Neo4j and Graph Data Science are uniquely well suited to:
  - Develop, curate, and manage a **high-quality grounding data set at scale**
  - **Log LLM interactions** with the grounding data
  - Visualize the LLM interactions and grounding data as a graph to **increase interpretability**
  - **Analyze LLM performance** to identify areas for improvement

Additional Resources

Slides, links to code repos, and links to blog articles are all published at:

https://github.com/danb-neo4j/NODES2023_GDS_GenAI

Thank you!

For more information or questions about grounding your LLM application with a knowledge graph, please contact us via sales@neo4j.com