# Developing a Natural Language Interface for Knowledge Graphs

Ruth Assefa, Sarah Mendoza
*Computer Science*
*Lyle School of Engineering*
Dallas, United States
rassef@smu.edu
sarahmendoza@smu.edu

Oyku Serap Ogut
*Computer & Data Science*
*Lyle School of Engineering*
Dallas, United States
oogut@smu.edu

Luke Voinov
*Electrical Computer Engineering*
*Lyle School of Engineering*
Dallas, United States
lvoinov@smu.edu

Dr. Nurcan Yuruk
*Computer Science*
*Lyle School of Engineering*
Dallas, United States
nyuruk@smu.edu

***Abstract*— This paper proposes to solve the challenge of making databases more user-friendly by interfacing them with OpenAI's ChatGPT-3.5 model. We implemented this solution to assist researchers in easily finding others with similar research interests. Our study involves 184 researchers from 14 departments at Southern Methodist University (SMU). We collected researchers' areas of expertise and biographies and stored them in a Neo4j graph database. We used OpenAI's embedding models to create vector representations of the collected data, allowing for accurate similarity assessments via Neo4j's built-in algorithms. By integrating this system with LangChain, we enabled natural language queries. The results demonstrated high accuracy in identifying related researchers and potential inter-departmental collaborations. The AI interfacing provided an easy way for users to interact with the database.**

***Keywords— graph database, knowledge graph, large language model, retrieval augmented generation, natural language, similarity search, semantic search, OpenAI, Neo4j, LangChain, User interfacing***

## I. Introduction

A common issue with software is that many problems require convoluted code which only experts understand. However, most of this software is distributed for non-experts to use. This can lead to difficulty in usage, as non-experts may have no idea how to navigate the software. Thus, even though the product can fulfill its purpose, most people will be unable to use it because they do not know where to start. We decided to address the problem of making a database, which falls into such a situation, more user-friendly through interfacing it with OpenAI's ChatGPT-3.5 model. The artificial intelligence (AI) model will allow users to query the database using natural language.

We decided to apply our interfacing on another problem: creating a user-friendly database that allows researchers to have an easy way of finding others who share similar research interests. We tackled this problem by gathering data and using a similarity algorithm on it. We collected data from 184 researchers from 14 departments in Southern Methodist University. This data was stored and organized as nodes in a Neo4j graph database. Each researcher node (if the data is not initially absent) contains a researcher's reported "research areas" and "biography"; these are the principal elements of data we use to assess researcher similarity. Using OpenAI's large and small embedding models, we created vector representations of each researcher's biographies and research areas. These embeddings were then added to their respective nodes, and we used a built-in Neo4j similarity algorithm to assess the similarity between nodes. We bridged the gap between the graph and ChatGPT through LangChain using Retrieval Augmented Generation (RAG) techniques.

After completing our model, we spent most of our time maximizing its accuracy. On the Neo4j side we experimented with which data yielded the most accurate results, and on the LangChain side we refined the model to retrieve and generate answers more accurately.

The benefits of the model are: 1) it quickly finds relationships (with high accuracy) that would take humans a long time to find, if ever; 2) it is adept at matching researchers from different disciplines; and 3) it is prompted via natural language, making it user friendly. As a bonus the AI model can find hidden associations between researchers that the Neo4j algorithm was unable to find.

## II. Methods

### *Process Overview*

In order to gather information about each professor's research interests, we extracted their information from their profiles located on their respective department's webpage. We extracted data from 14 different webpages, each representing a different department within SMU (Southern Methodist University). Half of them were from the Lyle School of Engineering and the rest were from the Dedman School of Humanities and Sciences. Information about education, research areas, biography, and recent publications were present for most of the professors we extracted. We decided that research areas and biographies were most representative of each professor's research interests. To extract this information, we sent HTTP requests through Python to the researcher webpages and used a python library, Beautiful

Soup, to parse the relevant data. The information was then saved in CSV files using the pandas software library. In total, we extracted the information of 184 researchers. See Appendix III for more details about the websites and specific departments.

We used Neo4j to create an accurate graph representation of the researchers by uploading the data we had parsed and saved in the CSV files. The nodes that we created using extracted information were the "University", "School", "Department", and "Researcher" nodes. Each node contained properties relevant to the node type, such as the "Researcher" nodes containing name, biography, research areas, and department properties. Nearly every node in the database had at least one relationship with another node, mainly for organizational purposes.

With the Neo4j graph finished, we used Python to extract each professor's information from our database to generate vector representations of their research areas and bios. The code runs a query in the Neo4j database that outputs each professor's name, id, and their biography or research area. The data is then run through an OpenAI API which creates the embeddings. Next, each professor's name and their embeddings were saved into separate CSV files. Once all the vector embeddings were added to the file, we uploaded them to the Neo4j database and assigned each embedding to their respective researcher.

*Creating a Neo4j Graph*

Neo4j, a graph database system, describes itself as follows:

"A Neo4j graph database stores nodes and relationships instead of tables or documents. Data is stored just like you might sketch ideas on a whiteboard. Your data is stored without restricting it to a pre-defined model, allowing a very flexible way of thinking about and using it." [1]

We used Neo4j for storing our data because it stores data efficiently, contains useful functions, allows for immediate and clear visualization, and is intuitive to navigate. Neo4j's original language Cypher is highly conceptual and intuitive.

The fundamental units of a Neo4j graph (or knowledge graph) are the node and the relationship. A Neo4j database can contain many complex structures, but ultimately, they are all built from nodes and relationships. Neo4j graphs were established upon ideas from graph theory [2], so graph theory terminology and concepts (e.g. directed, weighted) apply.

*Nodes*

Nodes in Neo4j hold data that represents some object. Nodes are represented as colored-in circles, where the color arbitrarily represents the type of node it is. Companies, people, cups, or planets can all be represented by nodes (Fig 1.1). The node of type "Department" will look and behave differently from the node of type "Researcher". Besides having a type, each node must have at least one unique property. Neo4j will automatically assign an id property if no properties are assigned to the node. A node usually contains more than one property, depending on necessity.

*Relationships*

Relationships show how nodes are connected to each other. They are represented as an arrow pointing from one node to another. Like nodes, they each have their own type (Fig. 1.1) and must have at least one property, which if not provided defaults to some id. Relationships can also hold as many properties - or in this context, weights - as necessary. In our situation mainly the nodes contained properties and the relations had none.

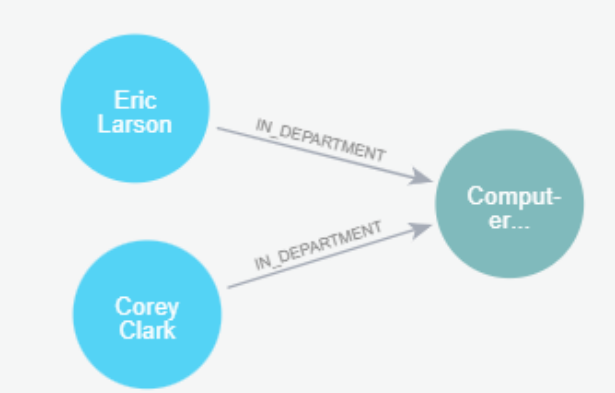


*Figure 1: The nodes "Eric Larson" (E) and "Corey Clark" (C) of type "Researcher" are distinct from each other, and distinct from "Computer Science" (CS) of type "Department". Furthermore, E and C contain properties such as "name" and "bio". E and C are not related to each other, but both are related to CS because of the relation "IN_DEPARTMENT". Notice that the relations are one way.*

*Schema of our Graph*

Our graph has 5 distinct nodes: Researcher, Department, School, University, and Query. It also has 3 distinct relationships: IN_DEPARTMENT, DEPARTMENT_OF, and IN_UNIVERSITY. Each of these nodes, except for the Query node, is connected via one of these relations (Fig. 1.2). Since all our researcher data was gathered from Southern Methodist University, the University node is redundant until we begin gathering data from other universities. The most important nodes are the Researcher and Query nodes. They are the ones being compared; all the other nodes serve an organizational purpose. Each Researcher node has multiple properties, the most relevant being the "name", "bio", "researchAreasString", "researchEmbeddingLARGE", and "bioEmbeddingLARGE" properties (see Appendix I for a full list of the properties each node type has). If the researcher did not report their research areas or biographies, their node contains neither of those properties nor the embeddings of those properties. The relations have no properties and serve an organizational purpose.

The data that Researcher nodes' properties contain is self-explanatory, e.g. "bio" contains the researcher's self-reported biography. There are a few things of note though. Confusingly, each Researcher contains a "researchAreasString" and "researchAreas" property. Initially we assigned Researchers the property "researchAreas", which is not a string but an array. "ResearchAreasString" is a second version of how we assigned a given researcher's self-reported research areas to their respective node. It is more convenient to find a researcher with a certain research area by finding an element in an array than to parse the string and extract the phrase. However, this array format made it more difficult to work with ChatGPT and we realized that parsing the data yields results more reliably than searching for an element. Other than the format, both properties have the exact same information. We have both versions because the array format is easier to quickly experiment with whereas the string format is more accurate.

The "researcher/bio EmbeddingLARGE" properties also deserve explanation. Each research node contains not just a "researcher/bio EmbeddingLARGE" property but also a "SMALL" version. Both versions of the properties are floating point arrays. The major difference between these properties is that the SMALL version's vector dimensions (1536) are less than the LARGE version's vector dimensions (3072). Our final model only runs a similarity search on the LARGE version for reasons discussed in the Results section.

The Query node is unique in the sense that it only interacts with the Researcher node. It only has two properties: a name (acting as an identifier) and a "queryEmbedding". The "queryEmbedding" is an embedding of some research interest. Our query nodes vary from "Deep Learning" to "Neuroscience". These words were run through an embedding process, which yielded their vector representations, with which we were able to semantically compare which research/bio embeddings were most like the query embedding. Without the query nodes we would be unable to find which researchers most align with some field and be limited to only finding which researchers most align with other researchers. Not mentioned previously, but present in the nodes, is the "BioAndAreas" property. This property is just a concatenation of biographies and research areas. If one of these was missing, we set "BioAndAreas" as whichever one was available, meaning a few nodes have two different properties with the exact same content. Our reasoning behind this was so as not to exclude researchers from being found when a similarity search was run on "BioAndAreaEmbeddingsLARGE". Many researchers who would otherwise fit some criteria well would not be considered just because they had one type of information and not the other.

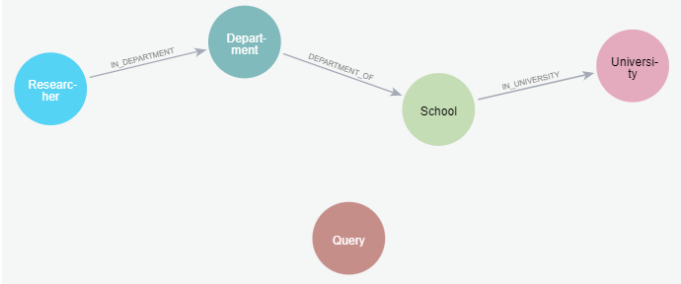


*Figure 2: In Cypher pseudocode: (Researcher) is [IN_DEPARTMENT] of (Department). (Department) is a [DEPARTMENT_OF] a (School). The (School) is [IN_UNIVERSITY] of (University). A specific example could be (Nurcan Yuruk) is [IN_DEPARTMENT] of (Computer Science) which is a [DEPARTMENT_OF] (Lyle) which is [IN_UNIVERSITY] of (SMU).*

*Embeddings Creation*

In machine learning, vectors are made up of floating-point numbers and used to represent words and phrases. These vectors are commonly referred to as embeddings because the numbers have context: meaning is "embedded" into them. OpenAI offers many text embedding models, but the two we used were text-embedding-3-small and text-embedding-3-large. The small embeddings have a dimension of 1536 while large embeddings have a dimension of 3072 [3]. The large embedding model is considered the better performing model since it scored a higher average on the multi-language retrieval benchmark (MIRACL) with a score of 54.9% compared to the small embedding model's score of 44%. The large embedding model also beats the small embedding model by 2.3% on the Massive Text Embedding Benchmark (MTEB). Although the large embedding model has high marks, the small embedding model is better for storage because it costs less to compute and store [4].
Embeddings exist in a latent space, which can be thought of as a multidimensional coordinate system. Each vector projects from the origin to some coordinate – this coordinate is the word. Similar words are generally grouped together. A simple example could be of a cartesian coordinate system where the x direction is "dog", and the y direction is "cat". A fox might have a higher dog component than cat component and thus would be further along the dog axis than cat axis. Fox would, by nature of the system, be closer to "jackal" than to "lion" [5]. Of course, latent space has many more dimensions than this example would suggest.
Embeddings are great at capturing semantic meaning – not only the word or phrase, but the underlying context behind it. For example, the word "apple" can be represented by the following vector (large embedding):

[-0.03898361697793007,0.029192212969064713,-0.0198558010160923,0.016825562343001366,-0.016015678644180298,-0.01217555534094572,-0.013895420357584953…]

If a word is semantically related to the word “apple”, its embedding is most likely to be similar to the embedding for “apple”. Table 1 contains a list of words that are similar to the word “apple” and their embeddings:

*Table 1: Word Embeddings*

| Word | Embedding |
|---|---|
| “iPhone” | [-0.0560863241553306 6,0.03206729143857956,-0.01888107694685459,0.013320950791239738,-0.020569775253534317…] |
| “fruit” | [-0.035383202135562 9,0.03866007179021835,-0.022118858993053436,-0.02159595489501953,-0.02448935993015766...] |
| “pie” | [-0.0291090477257967,-0.006916141603142023,-0.021155256778001785,0.013192323036491871,0.009124004282057285…] |
| “newton” | [-0.023355646058917046,0.00441383570432663,-0.01779395155608654,-0.0002205022901762277,-0.03194941580295563…] |
| “nutrition” | [0.007186924107372761,0.013432224281132221,-0.017255056649446487,-0.0011548976181074977,0.018349593505263 33…] |

Based on Neo4j’s semantic search – an algorithm that analyzes a search/query based on the context of a phrase rather than a summation of individual words – the embedding representing the word “apple” has more of a similarity to the embeddings representing the words “iPhone”, “fruit”, and “pie” than to the embedding representing “submarine”, for example. The embeddings for “newton” and “nutrition”, words which have less semantical similarity to “apple”, have less similar embeddings compared with the embedding representing “apple”.

*Similarity Search Algorithm*

Semantic search algorithms judge the similarity between words by comparing the embeddings and producing a similarity score [6]. The algorithm then ranks each similarity score from highest to lowest and returns the node associated with the score. In this process the algorithm matches the most relevant embeddings with a queried embedding (e.g. “iPhone” is matched with the queried word “apple”). Table 2 depicts the similarity scores of the words matched with “apple”:

*Table 2: Similarity Scores*

| Word | Score (rounded to fourth decimal) |
|---|---|
| “iPhone” | 0.8295 |
| “fruit” | 0.8141 |
| “pie” | 0.7580 |
| “newton” | 0.7330 |
| “nutrition” | 0.6540 |

The algorithm has multiple ways of gauging similarity, the most common ones being distance and angle, commonly called Euclidean and cosine similarity, respectively. The Euclidian method calculates similarity based on how far one node is from another by calculating the vector’s magnitude. The cosine method calculates similarity based on the cosine between two vectors, using the dot product [7]. OpenAI recommends using a cosine similarity function on its embeddings because it is slightly faster to compute [8]. While we did use cosine similarity for the algorithm, it is interchangeable with Euclidean similarity. The only difference is that the scores computed with cosine similarity were roughly 1.5 times greater than those computed with Euclidean. However, because the ordering of researchers is exactly the same for both similarity methods, the actual scores are arbitrary. What matters is not how high the similarity scores are but that the algorithm was able to determine which nodes are most similar to a queried node and, relative to the other scores, assign a high score to them.

We found our similarity scores using the following code:

```
CALL db.index.vector.queryNodes(nodeIndex,
numberOfNearestNeighbors, embedding)
```

What it did was run a cosine similarity search through every node inside an index, which is a subset of nodes. It is not obvious here, but cosine similarity was assigned to the index and the similarity search ran by the code above uses whatever similarity method the index assigns it [9].

*Methods for Testing*

We tested different metrics for matching researchers. Our main three metrics were “bio”, “researchAreas”, and “BioAndAreas”. Biographies are a metric used to capture a researcher’s educational and professional background and the research they have conducted, while the research areas are a metric used to identify the specific fields a researcher engages in. The combined version of the biography and research area captures both previous metrics and condenses them into one with the idea that more information is better. Using these metrics, we determined which one is most optimal for matching researchers.

Comparing the different metrics, we used various methods to find the researchers when creating the matching process. One method involved matching researchers to one specific researcher (name-based). We compared the selected researcher's research areas, biography, and combined metrics with every other researcher in our database to find which ones were most like the selected one. This method is important because it allows us to pinpoint the researchers who have similar research interests with a specific researcher and objectively ensures that the matching process makes logical sense.

Another method was finding researchers who have a certain phrase in their biography or research area. For example, if we want to find researchers who are most engaged in machine learning, the phrase "machine learning" would be specifically sought to find researchers whose interests most reflect machine learning. This method is the same as the previous one except it can find multiple researchers who fit the description (there can only be one researcher R but multiple researchers who study machine learning).
We also used a method in which we matched researchers based on the query node embeddings (field-based). As mentioned in the "Schematic of our Graph" section, the query nodes contain an embedding of a certain research field, such as deep learning. Using the embedding we compare it to the research areas, biographies, and combined embeddings to find researchers who are most involved in the field the query node represents. Where the phrase search method is rigid and works only if there is a phrase letter for letter in the metric, the query-based search is more flexible and associative – this search can find nodes that describe "machine learning" without using the exact term.

*Integration with OpenAI LLM*

After gathering our data and storing it in a Neo4j graph database, we used the Langchain framework to connect to OpenAI's GPT 3.5 model and implement a RAG system. This framework is accessible via a Python library called LangChain, and it simplifies building LLM applications by providing abstracted objects, functions, and operators. We made use of the library's chain, retriever, and agent features.

*Chains*
Chains are objects that contain a "sequence of calls" [10], where the output of one process is passed as input to the next. Chains are primarily defined using LangChain Expression Language (LCEL), and they can contain functions, LLM prompts, document retrievers, as well as LLM objects and agent executor objects.

*Retrievers*
Retriever objects serve as wrappers for vector stores. These objects can accept a text-formatted query, embed it, and run a similarity search by comparing the query's embedding against those in the vector store [11]. Upon identifying the documents with embeddings most like the query, the retriever will return the text data associated with the documents. In this project, we used retrievers to wrap Neo4jVector objects, which access our graph database to compare vector embeddings and retrieve text data from up to six relevant nodes.

*Agents and Tools*

Langchain documentation describes agents as "systems that use LLMs as reasoning engines to determine which actions to take and the inputs to pass them" [12]. As of Langchain version 0.2, agent objects can be initialized using an LLM API connection, an LLM prompt, and a list of tools available to the agent. The agent is then wrapped with an agent executor class, which can be invoked directly or called as part of a chain.

Broadly, tools are "interfaces that an agent may use to interact with the world" [13]. This includes interacting with other applications, such as searching the internet or accessing a database, but can include myriad other actions that inform the agent's final text output. Each tool is initialized with a name, a description, and a function that will be called upon the tool's execution. Both the name and description are represented as text data that an agent will use to decide whether it should use the tool or not. As such, these variables can be designed to include specific instructions and examples that tailor the agent's choice of tools and achieve results closer to ones that are desired or expected.

*RAG Architecture*
The major components of our RAG system can be split between its integration with OpenAI and Neo4j, and how it employs Langchain features to handle various query types.

*Connections to OpenAI and Neo4j*
The first section of our RAG implementation concerns establishing connections to OpenAI and the Neo4j database and initializing the retrievers that perform a search task. Langchain provides multiple classes that streamline connecting to various databases and language models, to include OpenAI and Neo4j.
We initialized our OpenAI connection using a ChatOpenAI object, where we provided an API key, selected the GPT 3.5 model, and set the model temperature to zero. The temperature of a language model indicates the amount of randomness applied while generating its responses. Since the goal of RAG is to produce grounded, factual answers, setting the OpenAI model's temperature to zero improves its accuracy and prevents hallucination. Additionally, using the GPT 3.5 model during the development stage allowed us to test our RAG system using a faster and less resource-intensive LLM. Each

of these settings are saved in the ChatOpenAI object, making it convenient to invoke the LLM or change its settings at a later time.

The Langchain library also provides various classes that can access Neo4j graph databases. We initialized our graph connection by storing the graph url and authentication information in a Neo4jGraph object. The graph object allowed us to run cypher queries on the graph from the application code, and to grant graph access to other objects by passing it as a parameter. We initialized two Neo4jVector objects to access the "researchEmbeddingLARGE" and "bioEmbeddingLARGE" data stored in our database. Finally, we used Langchain's retriever interface to wrap the Neo4jVector objects and set their standard number of returned documents to six. While vector stores can also perform similarity searches, wrapping them as retrievers fixes extraneous function parameters, so that different queries run with the same retriever object are executed with comparable settings.

*Query Handling*

To handle various types of natural language queries, we used a langchain agent to categorize queries based on whether they provide name-based information or field-based information. Examples of these query types are available in the results section, where queries 1-3 are name-based, and queries 4-6 are field-based.

On receiving a field-based query, the agent calls a tool that accepts string input. This input is formatted by extracting the research area descriptions from the query, separating them by commas if multiple apply, and passing them into the tool. The tool then calls a function to perform a retriever-based search with the input, and format the results to extract the researchAreasString, name, and department text data from returned documents. In this action, the tool's input format removes extraneous words from the query, thereby refining the vector embedding search.

Since name-based queries are highly useful, but do not provide any inherent information about research areas, they must be handled separately. On receiving a name-based query, the agent calls a custom tool that identifies a specific researcher in the graph and runs a similarity search using their text data. This tool accepts the original query string as input, and it consists of an LLM call, two functions, and a retriever call. The LLM call compares the query with a list of names that exist in the graph and extracts the in-graph name that most closely matches the query. This name is passed to a function that queries the graph to retrieve the particular researcher's researchAreasString property. Finally, the researchAreasString property is passed into a retriever call, and the output is formatted before returning the search results to the agent. This process closely mirrors the search steps performed by the database queries, where a specific node is identified, and a similarity search is performed using its existing properties.

## III. Results

*Query Examples*

To evaluate the performance of our finalized RAG model, we designed six queries that would test its ability to identify researchers who specialize in fields relevant to a user's question. These test queries are as follows:

1. Which professors have similar research interests as Eric Larson?
2. Which professors have similar research interests as Mehak Gupta?
3. Which professors have similar research interests as Nurcan Yuruk?
4. Which professors have deep learning as a research interest?
5. Which professors have natural language processing as a research interest?
6. Which professors have neuroscience as a research interest?

We designed these queries with an emphasis on Computer Science (CS) researchers and interdisciplinary fields. Since we are most familiar with research areas in CS, focusing on CS faculty members improves our ability to verify the quality of our search results by comparing the lists of research areas associated with each professor. As such, we selected the CS department researchers Eric Larson, Mehak Gupta, and Nurcan Yuruk as query subjects. We designed the field-based queries according to similar logic, with an increased focus on interdisciplinary fields. Deep learning and natural language processing are primarily CS fields, but their connection to the subject of artificial intelligence makes them strong candidates for interdisciplinary research in fields such as mathematics, statistics, and cognitive science. Neuroscience is also highly interdisciplinary, as it combines fields including biology, psychology, and even computer science to study the human nervous system. The multidisciplinary nature of these fields allows our queries to demonstrate the RAG model's ability to identify similar research areas across seemingly disparate departments.

In the following sections, results are discussed individually for each of the above queries. Research area text data is provided on a case-by-case basis, and a comprehensive list of research areas for the faculty members noted in the results section may be viewed in Appendix II.

*Similar Research as Eric Larson*

The first query produced similar results from the Cypher queries and RAG model. There is a 4/5 similarity between the two sets

of results, as both search techniques identified Corey Clark, Mehak Gupta, Frank Coyle, and Michael Hahsler as having similar research interests as Eric Larson. The two unique results were Scott Douglas, identified by the Cypher query, and Klyne Smith, identified by the RAG model. Since Scott Douglas is part of the Electrical and Computer Engineering department, the Cypher query performed slightly better in identifying faculty members from different disciplines. However, the 4/5 similarity between the two techniques indicates that the RAG model produces a comparable quality of results relative to a direct database query.

*Table 3: Q1 Cypher and RAG Results*

| Cypher Query Output | | | RAG Retriever Matches | |
|---|---|---|---|---|
| **Names** | **Department** | **Score** | **Name** | **Department** |
| Corey Clark | Computer Science | 0.8294 | Corey Clark | Computer Science |
| Frank Coyle | Computer Science | 0.8115 | Mehak Gupta | Computer Science |
| Michael Hahsler | Computer Science | 0.8071 | Frank Coyle | Computer Science |
| Scott Douglas | Electrical Computer Engineering | 0.7993 | Michael Hahsler | Computer Science |
| Mehak Gupta | Computer Science | 0.7825 | Klyne Smith | Computer Science |

**RAG Text Output: "**Professors with similar research interests as Eric Larson include Corey Clark, Mehak Gupta, Frank Coyle, Michael Hahsler, and Klyne Smith."

*Similar Research as Mehak Gupta*

The second query produced a 3/5 similarity between the Cypher query results and RAG model output, with the RAG model performing slightly better in accuracy and in identifying researchers from separate departments. Both search methods identified Labiba Jahan, Michael Hahsler, and Eric Larson as having similar research interests to Mehak Gupta. However, while the Cypher query also identified Kasilingam Periyasamy and Nurcan Yuruk, the RAG model instead identified the professors Jing Cao from the Statistics and Data Science department, and David (King Ip) Lin from the CS department.

*Table 4: Q2 Cypher and RAG Results*

| Cypher Query Output | | | RAG Retriever Matches | |
|---|---|---|---|---|
| **Names** | **Department** | **Score** | **Name** | **Department** |
| Labiba Jahan | Computer Science | 0.8294 | Labiba Jahan | Computer Science |
| Kasilingam Periyasamy | Computer Science | 0.8115 | Eric Larson | Computer Science |
| Michael Hahsler | Computer Science | 0.8071 | Jing Cao | Statistics and Data Science |
| Eric Larson | Computer Science | 0.7993 | Michael Hahsler | Computer Science |
| Nurcan Yuruk | Computer Science | 0.7825 | David (King Ip) Lin | Computer Science |

**RAG Text Output: "**Labiba Jahan and David (King Ip) Lin are professors with research interests like Mehak Gupta."

Each of the four unique search results contain fields relevant to Mehak Gupta's research interests. However, the RAG model's results are both more interdisciplinary and slightly more relevant. The RAG model identified Jing Cao from the Statistics and Data Science department, and David (King Ip) Lin from the CS department. Since Mehak Gupta is from the CS department, Jing Cao is a strong interdisciplinary result. Additionally, they both have research interests in "machine learning," as well as similar interests in "medical data" and "clinical trial design". David (King Ip) Lin is also a strong match, as he shares the "natural language processing" interest with Mehak Gupta. While Nurcan Yuruk and Kasilingam Periyasamy are valid results, as their respective interests in "healthcare applications" and "applied machine learning" align with Mehak Gupta's research areas, these similarities are less specific than those identified by the RAG model's retriever output. Overall, both the Cypher and RAG searches produced accurate results for Q2, with the RAG model's output being slightly more relevant.

*Table 5: Q2 Research Areas Comparison*

| Names | Research Areas |
|---|---|
| Mehak Gupta | Deep Learning, Supervised and Semi-supervised machine learning, Structured and Unstructured medical data, Predictive modeling, Natural Language Processing |
| Kasilingam Periyasamy | Software Engineering, Formal Methods, Data Science, Healthcare Applications |
| Nurcan Yuruk | Data Mining, Social Network Analysis, Applied Machine Learning |
| Jing Cao | Clinical trial design and analysis, Bayesian methods and applications, Machine learning and text mining |
| David (King Ip) Lin | Natural Language Processing, Database Systems, Data mining |

*Similar Research as Nurcan Yuruk*

The third query also produced a 1/5 similarity between the Cypher query results and the RAG model results. The RAG model identified more specific matches and a greater number of interdisciplinary results.

*Table 6: Similarity Search Table for Nurcan Yuruk*

| Cypher Query Output | | | RAG Retriever Matches | |
|---|---|---|---|---|
| **Names** | **Department** | **Score** | **Name** | **Department** |
| Maya El Dayeh | Computer Science | 0.7976 | Michael Hahsler | Computer Science |
| Michael Hashler | Computer Science | 0.7923 | David (King Ip) Lin | Computer Science |
| Mehak Gupta | Computer Science | 0.7818 | Jing Cao | Statistics and Data Science |
| Labiba Jahan | Computer Science | 0.7741 | Chul Moon | Statistics and Data Science |
| Jia Zhang | Computer Science | 0.7729 | Eric Larson | Computer Science |

**RAG Text Output: "**Michael Hahsler, David (King Ip) Lin, and Eric Larson have similar research interests as Nurcan Yuruk."

Both the Cypher query and the RAG model produced relevant results, with the former's output emphasizing the machine learning field, while the latter's output focused on data mining and network analysis. In the Cypher output, 4/5 results included "machine learning" in their research areas, while the RAG model had 3/5 results that contained the phrase. Meanwhile, only 2/5 Cypher results included "data mining", compared to 4/5 RAG results that contained "data mining" or "text mining" in the research areas. Given that machine learning is a more popular research interest than data mining, results that contain data mining can be considered more relevant to Nurcan Yuruk's stated research areas.

Additionally, the RAG model produced more diverse results than the Cypher query search. While all 5 researchers identified by the Cypher query are in the CS department, the RAG model returned 3 CS faculty and 2 Statistics and Data Science faculty. These researchers, Jing Cao and Chul Moon, are both strong interdisciplinary matches to Nurcan Yuruk's research areas. Jing Cao's research interests include "machine learning and text mining," while Chul Moon's interests include "network analysis", each of which directly overlap with Nurcan Yuruk's research areas. Additionally, of the 9 total researchers identified by both search methods, Chul Moon is the only result to include "network analysis" research area. As such, the RAG model's identification of Jing Cao and Chul Moon demonstrates its ability to find similarities in research between different departments.

*Research in the Deep Learning Field*

The Cypher query search and RAG model performed similarly well in identifying researchers in the Deep Learning field. Of the 6 researchers identified by the RAG model, all 6 were also identified in the Cypher query's 10 results, with a 5/6 similarity between the top 6 results from each search. In these top 6, the Cypher query identified 4 researchers from the CS department, and 2 from outside the CS department, while the RAG model identified 5 researchers from the CS department, and 1 from a separate department. In particular, the Mathematics department researcher Difeng Cai was matched by both the Cypher query and the RAG model. His research areas are an accurate interdisciplinary match, as they specifically include "Studying mathematical structures in deep learning models for scalable computation". Altogether, both the Cypher query and RAG model found relevant and interdisciplinary results for Q4.

The additional four results from the Cypher query further demonstrate the ability of embedding-based similarity search to identify interdisciplinary results. In total, the database query matched 6 CS department researchers, 2 Mathematics department researchers, and 2 researchers from the Operations Research and Engineering Management department.

*Table 7: Similarity Search Table for Deep Learning*

| Cypher Query Output | | | RAG Retriever Matches | |
|---|---|---|---|---|
| Names | Department | Score | Name | Department |
| Mehak Gupta | Computer Science | 0.7232 | Mehak Gupta | Computer Science |
| Eric Larson | Computer Science | 0.7140 | Eric Larson | Computer Science |
| Frank Coyle | Computer Science | 0.6980 | Frank Coyle | Computer Science |
| Difeng Cai | Mathematics | 0.6954 | Michael Hahsler | Computer Science |
| Labiba Jahan | Computer Science | 0.6854 | Labiba Jahan | Computer Science |
| Miju Ahn | Operations Research and Engineering Management | 0.6789 | Difeng Cai | Mathematics |
| Wei Cai | Mathematics | 0.6773 | -- | -- |
| Corey Clark | Computer Science | 0.6743 | -- | -- |
| Digvijay Boob | Operations Research and Engineering Management | 0.6722 | -- | -- |
| Michael Hahsler | Computer Science | 0.6715 | -- | -- |

*Research in Natural Language Processing*

The results between the Cypher query and the RAG output have an 8/10 similarity. Both identified David (King Ip) Lin, Labiba Jahan, Frank Coyle, Mehak Gupta, Joshua Oltmanns, Scott Douglas, Michael Hahsler, and Nurcan Yuruk. Joshua Oltmanns, a professor in the Psychology department, is a good interdisciplinary result here since the majority of the professors output primarily study engineering, or more specifically computer science. This shows the diversity of the RAG model and Cypher query's ability to find relevant matches across different fields of study.

*Table 8: Q5 Cypher and RAG Results*

| **Cypher Query Output** | | | **RAG Retriever Matches** | |
|---|---|---|---|---|
| **Names** | **Department** | **Score (rounded to fourth decimal)** | **Name** | **Department** |
| David (King Ip) Lin | Computer Science | 0.7232 | Labiba Jahan | Computer Science |
| Labiba Jahan | Computer Science | 0.7140 | David (King Ip) Lin | Computer Science |
| Frank Coyle | Computer Science | 0.6980 | Frank Coyle | Computer Science |
| Sarah Kucker | Psychology | 0.6954 | Michael Hahsler | Computer Science |
| Mehak Gupta | Computer Science | 0.6854 | Mehak Gupta | Computer Science |
| Joshua Oltmanns | Psychology | 0.6789 | Nurcan Yuruk | Computer Science |
| Scott Douglas | Electrical Computer Engineering | 0.6773 | Joshua Oltmanns | Psychology |
| Eric Larson | Computer Science | 0.6743 | Scott Douglas | Electrical Computer Engineering |
| Michael Hahsler | Computer Science | 0.6722 | Eric Larson | Computer Science |
| Nurcan Yuruk | Computer Science | 0.6715 | Sarah Kucker | Psychology |

**RAG Text Output:** "Professors at SMU who have natural language processing as a research interest include Labiba Jahan, David (King Ip) Lin, Frank Coyle, Michael Hahsler, Mehak Gupta, Nurcan Yuruk, Joshua Oltmanns, and Eric Larson."

The results between the query node for Natural Language Processing (NLP) and the query node for Natural Language Processing with a focus on Computer Science and Machine Learning have a 9/10 similarity. Despite the results being nearly identical, the results dramatically shift between the two. For the NLP node, the top 7 researchers are diverse with 4 of them in the computer science department, 2 in the psychology department, and 1 in electrical computer engineering. Meanwhile, for the NLP node with focus, all the top 7 researchers are in the computer science department. Joshua

Oltmanns was previously in the top 7 researchers for the NLP query, but he lands in last for the top 10 researchers for the NLP with focus query. This is most likely because the NLP with focus node emphasizes "computer science" leading to less diverse results.

*Table 9: NLP vs. NLP with a focus on CS & ML*

| Query Node: Natural Language Processing | | | Query Node: NLP with a focus on Computer Science and Machine Learning | | |
|---|---|---|---|---|---|
| **Names** | **Department** | **Score** | **Names** | **Department** | **Score** |
| David (King Ip) Lin | Computer Science | 0.7232 | Labiba Jahan | Computer Science | 0.7711 |
| Labiba Jahan | Computer Science | 0.7140 | Frank Coyle | Computer Science | 0.7691 |
| Frank Coyle | Computer Science | 0.6980 | David (King Ip) Lin | Computer Science | 0.7621 |
| Sarah Kucker | Psychology | 0.6954 | Eric Larson | Computer Science | 0.7273 |
| Mehak Gupta | Computer Science | 0.6854 | Mehak Gupta | Computer Science | 0.7244 |
| Joshua Oltmanns | Psychology | 0.6789 | Michael Hahsler | Computer Science | 0.7118 |
| Scott Douglas | Electrical Computer Engineering | 0.6773 | Nurcan Yuruk | Computer Science | 0.7063 |
| Eric Larson | Computer Science | 0.6743 | Corey Clark | Computer Science | 0.7054 |
| Michael Hahsler | Computer Science | 0.6722 | Scott Douglas | Electrical Computer Engineering | 0.7031 |
| Nurcan Yuruk | Computer Science | 0.6715 | Joshua Oltmanns | Psychology | 0.6978 |

*Research in the Neuroscience field*

The Cypher query and RAG model searches produced highly similar outputs for Q6. The top 6 results for each search have a 5/6 similarity, and all 6 researchers identified by the RAG model were also matched in the 10 results from the Cypher query. The top 6 RAG retriever results included 3 mathematics department researchers, 2 biological science department researchers, and 1 in the electrical computer engineering department. The Cypher query resulted in 3 researchers in mathematics, 3 in psychology, 2 in biological sciences, 2 in electrical computer engineering. As such, both searches display capabilities of identifying similar research across separate departments, with strong consistency in results between the different search methods.

The Cypher query results for Holly Bowen and Prasanna Rangarajan both demonstrate the effectiveness of using embedding-based semantic search to identify similar research areas. Holly Bowen's research areas include "Memory, Emotion, Motivation, Aging" and "fMRI". Although her research areas do not include the phrase "neuroscience", she was still matched by the embedding search. This demonstrates the semantic search's ability to detect similarities based on contextual meaning, which gives both of our search methods greater flexibility relative to traditional keyword search models. Likewise, Prasanna Rangarajan's research areas do not include the keyword "neuroscience". However, her interests in "Neuromorphic Sensing, Optical Engineering", and "Computer Vision", each relate to computational and engineering aspects of the neuroscience field. By comparing research fields based on their context and semantic meaning, the Q6 Cypher query identified meaningful interdisciplinary similarities that would have been overlooked by a keyword search. Although the RAG model's results did not include Holly Bowen or Prasanna Rangarajan, it is likely that they would appear in a larger batch of retrieval results, given the demonstrated consistency between the Cypher query and RAG output for the neuroscience focused query.

*Table 10: Research Results for Neuroscience Field*

| Cypher Query Output | | | RAG Retriever Matches | |
|---|---|---|---|---|
| **Names** | **Department** | **Score** | **Name** | **Department** |
| Leslie-Ann Asmus | Mathematics | 0.7154 | Leslie-Ann Asmus | Mathematics |
| Zhihao Wu | Biological Sciences | 0.7101 | Kathryn Hedrick | Mathematics |
| Kathyrn Hedrick | Mathematics | 0.7049 | Zhihao Wu | Biological Sciences |
| Andrea Barreiro | Mathematics | 0.6918 | Andrea Barreiro | Mathematics |
| Amy L. Brewster | Biological Sciences | 0.6871 | Amy L. Brester | Biological Sciences |
| Holly Bowen | Psychology | 0.6781 | Carlos E. Davila | Electrical Computer Engineering |
| Carlos E. Davila | Electrical Computer Engineering | 0.6723 | – | – |
| Stephanie Wilson | Psychology | 0.6501 | – | – |
| Sarah Kucker | Psychology | 0.6408 | – | – |
| Prasanna Rangarajan | Electrical Computer Engineering | 0.6390 | – | – |

*Real Life Application*

*Inter-departmental Collaborations*

The developed system significantly enhances inter-departmental collaborations by identifying researchers with similar or complementary research interests across different departments. For example, a researcher in Computer Science may discover potential collaborators in the Psychology department who are investigating computational models of cognition. Utilizing this tool to identify these connections enables universities to come up with more interdisciplinary research projects.

*Research Networking*

Researchers often face challenges in identifying and connecting with colleagues outside their immediate professional circles. Our system provides an efficient method to locate and connect with researchers who share similar interests, even if they belong to different departments or schools. The system allows researchers to rapidly build a network of potential collaborators.

*Grant Applications*

Forming a diverse team with diverse backgrounds can substantially improve the chances of success in research grant applications. This system aids in identifying potential team members who can contribute different perspectives and expertise; therefore, strengthening the grant proposal. Also, it can be used to demonstrate the interdisciplinary nature of the proposed research, which is a key requirement for many funding agencies.

*Academic Events and Conferences*

Organizers of academic events and conferences can use this system to identify potential speakers or panelists with knowledge in specific areas. It can also help with forming discussion groups or workshops that bring together researchers from different disciplines to discuss common challenges.

## IV. Conclusion

This paper demonstrates the development of a natural language interface for knowledge graphs using OpenAI's ChatGPT-3.5 model. The model is integrated with a Neo4j database via the LangChain framework. This user-friendly system allows non-experts to query research data intuitively, addressing a common issue in database usability.

We evaluated the system's performance through six test queries that successfully identified professors with similar research interests. In the interdisciplinary research areas, our model demonstrates high accuracy and efficiency, identifying strong interdisciplinary connections that traditional keyword searches could not.

This system holds significant potential for enhancing research networking, improving the formation of interdisciplinary teams for grant applications, and assisting in organizing academic events and conferences. This approach not only improves accessibility and usability but also fosters greater collaboration and innovation within the academic community. Future steps will focus on improving system accuracy by considering researcher publications as a metric, expanding capabilities, adding more researchers from different universities, and exploring broader applications.

## References


[1] Neo4j, "What is a graph database? - Getting Started," *Neo4j Graph Data Platform*, 2024. https://neo4j.com/docs/getting-started/get-started-with-neo4j/graph-database/

[2] Neo4j, "Graph Thinking," *graphacademy.neo4j.com*. https://graphacademy.neo4j.com/courses/neo4j-fundamentals/1-graph-thinking/ (accessed Jul. 29, 2024).

[3] OpenAI, "OpenAI API Embeddings," *platform.openai.com*. https://platform.openai.com/docs/guides/embeddings

[4] OpenAI, "OpenAI Platform Embeddings," *platform.openai.com*. https://platform.openai.com/docs/guides/embeddings/

[5] K. Henner, "An intuitive introduction to text embeddings - Stack Overflow," *stackoverflow.blog*, Nov. 09, 2023. https://stackoverflow.blog/2023/11/09/an-intuitive-introduction-to-text-embeddings/

[6] Neo4j, "Semantic Indexes - Cypher Manual," *Neo4j Graph Data Platform*. https://neo4j.com/docs/cypher-manual/current/indexes/semantic-indexes/overview/ (accessed Jul. 30, 2024).

[7] Neo4j, "Vector Indexes - Cypher Manual," *Neo4j Graph Data Platform*. https://neo4j.com/docs/cypher-manual/current/indexes/semantic-indexes/vector-indexes/#similarity-functions (accessed Jul. 30, 2024).

[8] OpenAI, "OpenAI Embeddings FAQ," *platform.openai.com*. https://platform.openai.com/docs/guides/embeddings

[9] Neo4j, "Procedures - Operations Manual," *Neo4j Graph Data Platform*. https://neo4j.com/docs/operations-manual/5/reference/procedures/#procedure_db_index_vector_queryNodes (accessed Jul. 30, 2024).

[10] LangChain, "Chains | LangChain," *python.langchain.com*. https://python.langchain.com/v0.1/docs/modules/chains/

[11] LangChain, "Retrievers | LangChain," *python.langchain.com*. https://python.langchain.com/v0.1/docs/modules/data_connection/retrievers/

[12] LangChain, "Build an Agent | LangChain," *python.langchain.com*. https://python.langchain.com/v0.2/docs/tutorials/agents/#:~:text=A%20big%20use%20case%20for (accessed Jul. 30, 2024).

[13] LangChain, "Tools | Langchain," *js.langchain.com*. https://js.langchain.com/v0.1/docs/modules/agents/tools/#:~:text=Tools%20are%20interfaces%20that%20an (accessed Jul. 30, 2024).

## APPENDIX

### *Appendix A- 1: Node Properties*

| Node Type | Properties | Type | Relations |
|---|---|---|---|
| Researcher | researchAreasString<br>researchAreas<br>researchEmbeddingLARGE<br>researchEmbeddingSMALL<br>bio<br>bioEmbeddingLARGE<br>bioEmbeddingSMALL<br>BioAndArea<br>BioAndAreaEmbeddingLARGE<br>department<br>Id<br>name<br>new | STRING<br>LIST<br>LIST<br>LIST<br>STRING<br>LIST<br>LIST<br>STRING<br>LIST<br>LIST<br>STRING<br>STRING<br>BOOLEAN | IN_DEPARTMENT |
| Department | name | STRING | DEPARTMENT_OF |
| School | name | STRING | IN_UNIVERSITY |
| University | name | STRING | None |
| Query | name<br>queryEmbedding | STRING<br>LIST | None |

### *Appendix A- 2: Research Areas Text Data*

| Researcher | Research Areas |
|---|---|
| Miju Ahn | Mathematical optimization, Statistical learning, Nonconvex programming |
| Leslie-Ann Asmus | Mathematical and computational neuroscience, The study of dynamical, information theoretic and computational properties of neural networks, Fluid dynamics |
| Andrea Barreiro | Mathematical and computational neuroscience, including the study of dynamical, information theoretic and computational properties of neural networks, Fluid dynamics |
| Digvijay Boob | Design and Analysis of Algorithms, First-order methods, Variational Inequality and Minmax problems, Data-driven (stochastic) optimization, Machine Learning Theory |
| Holly Bowen | Memory, Emotion, Motivation, Aging, fMRI |
| Amy L. Brewster | Identify potential therapeutic targets for the prevention, treatment, and control of epilepsy, Identify whether neuro-immune interactions contribute to the construction of hyperexcitable neuronal networks that may promote seizures and cognitive deficits in epilepsy, Neuroscience, Inflammation and Immunity |
| Difeng Cai | Efficient algorithms for numerical linear algebra, partial differential equations and integral equations, Studying mathematical structures in deep learning models for scalable computation |
| Wei Cai | Developing advanced fast machine learning, stochastic, and deterministic numerical methods to understand complex physical phenomena with a special focus on quantum phenomena, electromagnetic processes, and wave scattering in random media |
| Jing Cao | Clinical trial design and analysis, Bayesian methods and applications, Machine learning and text mining |
| Corey Clark | Human-in-the-loop Machine Learning, Video Games and Artificial Intelligence, Distributed and Privacy Computing, Human and Machine Collaboration, Bias and Explainability in Artificial Intelligence, Game Based Education |
| Frank Coyle | Machine Learning / Artificial Intelligence, Software Engineering, Large Language Models |
| Carlos E. Davila | Biomedical and statistical signal processing, Brain-computer interfacing, Computational neuroscience |
| Scott Douglas | Audio, Array Processing, Machine Learning, Remote Sensing, Speech Enhancement |
| Maya El Dayeh | Cloud Computing, Data Mining, Software Engineering |
| Mehak Gupta | Deep Learning, Supervised and Semi-supervised machine learning, Structured and Unstructured medical data, Predictive modeling, Natural Language Processing |
| Michael Hahsler | Artificial Intelligence, Machine Learning, Data Mining, Data Science |
| Kathryn Hedrick | Scientific computation, Neuroscience |
| Labiba Jahan | Natural Language Processing, Narrative Understanding, Machine Learning |
| Sarah Kucker | Cognitive Development, Language Acquisition, Digital Media, Word Learning, Categorization |
| Eric Larson | Machine Learning and Deep Learning, Privacy and Security, Applied Machine Learning in Health and Education, Human Machine Teaming and Biometrics |
| David (King Ip) Lin | Natural Language Processing, Database Systems, Data mining |
| Chul Moon | Topological Data Analysis, fMRI Data Analysis, Network Analysis, Geophysical Statistics, Forensic Statistics |
| Joshua Oltmanns | Clinical and Personality Assessment, Clinical Psychology, Psychometrics, Structural Equation Modeling, Natural Language Processing, Artificial Intelligence |
| Kasilingam Periyasamy | Software Engineering, Formal Methods, Data Science, Healthcare Applications |
| Prasanna Rangarajan | Computational Imaging, Neuromorphic Sensing, Optical Engineering, Photonics, Computer Vision |
| Klyne Smith | Software Engineering Life Cycle Model Delivery Engagement, Project Management Optimization and Risk Management, Intersection and Dependencies of Software Engineering and Project Management, Requirements and Software Engineering associated with Cybersecurity, Machine Learning, and Artificial Intelligence |
| Stephanie Wilson | Family relationships, Social interaction, Psychoneuroimmunology, Adulthood and aging, Autonomic function, Ecological momentary assessment |
| Zhihao Wu | Neuroscience and Gene Regulation, Gene Regulation |
| Nurcan Yuruk | Data Mining, Social Network Analysis, Applied Machine Learning |
| Jia Zhang | Data Science Infrastructure, Machine Learning, Scientific Workflow, Software Engineering, Cloud Computing |

*Appendix A- 3: Department Statistics and Research*

| Department | # of Researchers | Percentage Total (%) | Researchers | School |
|---|---|---|---|---|
| Computer Science | 20 | 10.6 | https://www.smu.edu/lyle/departments/cs/people/faculty | Lyle |
| Civil and Environmental Engineering + Sustainability and Development | 11 | 5.85 | https://www.smu.edu/lyle/departments/cee/faculty + https://www.smu.edu/lyle/departments/multidisciplinary-programs/ma-in-sustainability-and-development | Lyle |
| Electrical Computer Engineering | 18 | 9.57 | https://www.smu.edu/lyle/departments/ece/people/faculty | Lyle |
| Mechanical Engineering | 11 | 5.85 | https://www.smu.edu/lyle/departments/me/people/faculty | Lyle |
| Operations Research and Engineering Management | 12 | 6.28 | https://www.smu.edu/lyle/departments/orem/people/faculty | Lyle |
| Design and Innovation | 6 | 3.26 | https://www.smu.edu/lyle/departments/multidisciplinary-programs/madi/people | Lyle |
| Biological Sciences | 11 | 5.85 | https://www.smu.edu/dedman/academics/departments/biological-sciences/people | Dedman |
| Earth Sciences | 18 | 9.57 | https://www.smu.edu/dedman/academics/departments/earth-sciences/people | Dedman |
| Mathematics | 23 | 12.2 | https://www.smu.edu/dedman/academics/departments/math/people | Dedman |
| Physics | 13 | 6.91 | https://www.physics.smu.edu/web/people/ | Dedman |
| Statistics and Data Science | 14 | 7.45 | https://www.smu.edu/dedman/academics/departments/statistics/people | Dedman |
| Sociology | 11 | 5.85 | https://www.smu.edu/dedman/academics/departments/sociology/people | Dedman |

Total number adds up to 188 researchers because some fields share researchers. Percentages may not add up to 100% due to rounding