GEORGE MASON UNIVERSITY

Data Analytics Engineering

Fall 2022

GMU Faculty and Researcher Experts List

Mehrdad Ghyabi
Derek Margulies
Karthik Munnangi
Harshith Vangoor
Ben Zevin

DAEN 690 Project Report

George Mason University

12/9/2022

About the Cover

George Mason University's Arlington Campus has become Mason Square, a dynamic, collaborative hub uniting scholars, students, researchers, policymakers, and business developers that will accelerate the pace of change and solve grand challenges.

Situated within the Rosslyn-Ballston corridor the campus, minutes from the Virginia Square-GMU Metrorail station, consists of Vernon Smith Hall, Van Metre Hall, and Hazel Hall. In addition to classroom space, the campus site hosts the Antonin Scalia Law School, the Schar School of Policy and Government, the Jimmy and Rosalynn Carter School for Peace and Conflict Resolution, programs in Arts Management and the School of Business, and Continuing and Professional Education.

The heart of Mason Square will be FUSE (see cover rendering). With an estimated construction completion date of December 31, 2025, FUSE is a technology-forward building that will integrate classrooms, labs and workspaces, start-ups, and public programming. FUSE will promote the exchange of new ideas and the transfer of intellectual property between Mason faculty and students, university partners, the tech industry, and the community.

With nearly 350,000 square feet, FUSE at Mason Square will be a net-zero carbon emissions-ready building with:

- LEED Platinum, FitWel 2.1, and RELi 2.0 certifications
- Energy-efficient green roof for heating and cooling,
- An agile floor design to enable responsive team or project growth,
- Specialized labs for robotics, VR, simulation, security, and data visualization.
- A 50-seat theater-style multipurpose room,
- Double height atrium and outdoor plaza for retail and community events

Table of Contents

Abstract 5

1 Introduction 6

1.1 Background 6

1.2 Problem Space 6

1.3 Research 7

1.3.1 Previous Research: Recommending Experts Using Knowledge Graphs 7

1.3.2 Web Scraping 9

1.3.3 Natural Language Processing (NLP) 10

1.3.4 Graph Database 10

1.3.5 Neo4j 11

1.3.6 NeoDash 12

1.4 Solution Space 12

1.5 Project Objectives 14

1.6 Primary User Stories 15

1.7 Product Vision - Sample Scenarios (why would someone want to use this) 15

Scenario #1 15

Scenario #2 15

Scenario #3 15

1.8 Definition of Terms 16

2 Data Acquisition 18

2.1 Overview 18

2.2 Field Descriptions 19

2.3 Data Context 20

2.3.1 George Mason University College of Engineering and Computing Website 20

2.3.2 LinkedIn 21

2.3.3 Google Scholar 21

2.3.4 Academic Search Complete 22

2.4 Data Conditioning 22

2.4.1 George Mason University College of Engineering and Computing Website 22

2.4.2 LinkedIn 22

2.4.3 Google Scholar 23

2.4.4 Academic Search Complete 23
2.5 Data Quality Assessment 24
2.5.1 George Mason University College of Engineering and Computing Website 24
2.5.2 LinkedIn 25
2.5.3 Google Scholar 27
2.5.4 Academic Search Complete 28
2.5.5 GMU Library Portal 28
2.6 Other Data Sources 29
2.6.1 NIST and NSF 29
2.6.2 Taxonomy 29
2.6.3 Removed Sources 30
2.7 Data Collection Ethics 31
3 Analytics and Algorithms 32
3.1 Web Scraping 32
3.1.1 College of Engineering and Computing Website 33
3.1.2 LinkedIn 34
3.1.3 Google Scholar 36
3.2 Preprocessing and Consolidating Data 37
3.2.1 Preprocessing 37
3.2.2 Consolidating Datasets 37
3.3 Natural Language Processing (NLP) 38
3.3.1 Bag of Words 38
3.3.2 Associating researchers with research fields 39
3.3.3 Vectorization and Cosine Similarity 39
3.4 Graph Database Implementation 40
4 Visualization 45
4.1 Cosine Similarity Findings 45
4.1.1 Frequency Visualizations 45
4.1.2 Word Cloud Visualizations 47
4.1.3 Score Distribution Visualizations 48
4.2 NeoDash Dashboards 50
4.2.1 Database Schema Dashboard 50
4.2.2 Category Search Dashboard 50

4.2.3 Area Search Dashboard 51
4.2.4 Name Search Dashboard 52
4.2.5 Document Search Dashboard 53
5 Findings 56
6 Summary 57
7 Future Work 58
8 Appendix A: Code References 60
8.1 Link 60
8.2 Project Description 60
8.3 Project Goals 60
8.4 Project Deliverables 60
8.5 Repository Contents 61
9 Appendix B: Installation Guide 62
9.1 Neo4j and NeoDash 62
10 Appendix C: Risk Section 73
10.1 Sprint 1 Risks 73
10.2 Sprint 2 Risks 75
10.3 Sprint 3 Risks 77
10.4 Sprint 4 Risks 79
10.5 Sprint 5 Risks 80
11 Appendix D: Agile Development 81
11.1 Scrum Methodology 81
11.2 Sprint 1 Analysis 82
11.3 Sprint 2 Analysis 82
11.4 Sprint 3 Analysis 83
11.5 Sprint 4 Analysis 83
11.6 Sprint 5 Analysis 84
12 References 85

Table of Figures

Figure 1. Front-end interface design for recommending research advisors. From Rahdari and Brusilovsky. 8
Figure 2. Knowledge Graph Example [4] 9
Figure 3. Graph Database example [14] 111
Figure 4. Solution Space Diagram 14
Figure 5. Six highly cited Google Scholar profiles affiliated with GMU 182
Figure 6. Sample Google Scholar Profile 193
Figure 7. Available publication data on Google Scholar 215
Figure 8. CEC Website Profile Example 344
Figure 9. LinkedIn About Section Example 35
Figure 10. LinkedIn Publication Example 366
Figure 11. Multiple Authors in Publications 377
Figure 12. Consolidated Dataset 38
Figure 13. Bag of Words. 39
Figure 14. Snapshot of similar categories of each article along with the scores 40
Figure 15. Taxonomy Before "Explode" 411
Figure 16. Taxonomy After "Explode" 422
Figure 17. Database Person Object 433
Figure 18. Database Relationships 444
Figure 19. Person Category Label Frequency 466
Figure 20. Person Area Label Frequency 466
Figure 21. Person Category Word Cloud 47
Figure 22. Person Area Word Cloud 48
Figure 23. Person Category Score Distribution 49
Figure 24. Person Area Score Distribution 49
Figure 25. Database schema graph 500
Figure 26. Category search dashboard for machine learning 511
Figure 27. Area search dashboard for software engineering 522
Figure 28. Name search dashboard for Dr. Vadim Sokolov 533
Figure 29. Document Search Dashboard Part 1 544
Figure 30. Document Search Dashboard Part 2 556

Abstract

George Mason University (GMU) is one of only 146 institutions in the United States that is classified as an R1 University: a university recognized as one with very high research activity. To maintain R1 University status, finding relevant research partners within GMU for research projects, proposals, and grants efficiently is of utmost importance. Currently, GMU is lacking a tool that is able to find current research partners with up-to-date relevant experience. Two researchers at the University of Pittsburgh addressed a similar problem and developed knowledge graphs to help students connect with potential research advisors. A readily available database containing faculty member subject expertise and conducted research would significantly reduce the time researchers spend looking for research partners. This project consists of four steps: data collection, natural language processing, data storage, and visualization/reporting. Data cleaning is completed using self-built web scrapers. Natural language processing uses cosine similarity to compare concepts of research areas to the biographies and abstracts of papers published by faculty members. Data storage uses graph databases powered by Neo4j. Finally, visualizations/reporting use NeoDash to create dashboards to display data from the database. Our team created a graph database prototype that connects faculty members to prospective research categories and areas and connects their published material to potential research subjects. We discovered that the stated research interests of faculty members are a much more reliable source of information, and the relationships developed using cosine similarity provide a less accurate option for potential research experts. The solution provides rudimentary search capabilities looking for collaborators; however, it is currently limited by data made available online. We believe it has the potential to grow into an important tool in GMU's desire to continue being one of the country's top research institutions.

1 Introduction

1.1 Background

The C4I & Cyber Center at George Mason University (GMU) is the nation's first and only civilian university-based entity [1]. The Center offers a comprehensive academic and research program in military-based information technology (IT) and cyber security applications [1]. The mission of this Center is to perform advanced research in defense, intelligence, and security-related applications in Information technology and cyber security. In addition, they focus on bridging cultural gaps and aligning government, industry, and academic requirements [1].

The C4I & Cyber Center wants to be the central location for all information and research conducted within the IT and Cyber world. Furthermore, it wants to connect faculty and researchers with interests in the Center's mission and to be recognized as a significant source of knowledge and development in military and civilian authorities [1].

The Center is working in various research areas, including C4 architectures, command support and intelligent systems, and modeling and simulation [1]. Some of the research currently being conducted by the Center include a semantic testbed for inference enterprise multi-modeling, which uses available information about an inference to predict the performance of the enterprise [1]. Another project they are working on is a SmartNode Pod designed to establish battlefield interoperability and be flown on either manned or unmanned aircraft to establish a hub that will increase ranges for communications and networks [1]. As a result, the Center has become a significant research contributor to the intelligence community. Furthermore, the Center conducts active outreach programs to government and industry and is a major contributor to NATO, AFCEA, STIDS, and ICRRTS conferences [1].

While the organization requesting the creation of a GMU Faculty and Researcher Experts List is the C4I & Cyber Center, the point of contact for the Center and the person that brought the project to the program is Dr. Linton Wells. Dr. Wells has a background in science, engineering, and social science with 51 years of experience working for the defense department and has been at George Mason in the Center since 2016. The origin of this project was born from his frustration with trying to write transdisciplinary proposals for National Science Foundation (NSF) grants, government grants, and other types of grants. For example, Dr. Wells found it very difficult to apply AI to the development of power grids in Appalachia, so he tried to find a subject expert in George Mason, which proved difficult. Since he started working at George Mason, this same frustration has occurred 10 to 15 for various projects. This frustration caused Dr. Wells to go to the Data Analytics Engineering Program to look for possible solutions to this issue.

1.2 Problem Space

A public research university with campuses in South Korea and Virginia, GMU comprises more than ten separate schools. Since George Mason is a public research institution, all its schools are conducting extensive research on a variety of subjects, from humanities studies to cyber security. The University also receives millions of dollars annually to perform research that will benefit this nation and the rest of the world. As a result, all the resources at hand must be used to undertake the research. This entails collaborating with coworkers from several GMU departments with diverse professional experiences.

The focus of the problem is that a researcher is searching for a data analytics engineering solution to assist him in finding suitable research partners when submitting a grant or project proposal.

At GMU, each department has a sizable number of academic staff members who do research on a variety of topics. Each department has its own website with a list of the professors' ongoing research projects, information about their research specialties, and opinions on their own work. One of the main problems of this project is that all the information presented on the department's websites may not be up to date. Each academic member also has a listing of their interests, areas of specialization, and body of work on a variety of websites, such as Google Scholar, LinkedIn, and college websites. Unfortunately, this data is unorganized and only available in fragments. There is no centralized database where someone can search for an expert on the topic they are looking for and contact them for advice. One must visit the websites of several departments to locate the appropriate academic member with whom they might collaborate or to find the person conducting research on a related topic. Finding the perfect candidate for the position is like searching for a needle in a haystack. In order to contact the professor, one must look through all the departments' websites to find pertinent subject-matter experts. It takes a lot of effort to find the correct person by exploring every department's website and accumulating data.

The main objective of this project is to present a data analytics engineering solution for GMU that includes all the information on the academics and their pertinent work to make it simple for someone looking for a subject matter expert to find the right individual. We will first attempt to establish a database for one department, namely the College of Engineering and Computing (CEC) because combining the data of all the academics and their work from all the departments is a significant undertaking. It would be wise to start with the CEC department because it is a large department and many of its members work in other departments. Once this prototype has been made, the remaining departments can be added. This development will assist everyone looking for an expert's viewpoint on a specific issue as well as make it simple to connect with someone who shares their academic interests to collaborate and work on research.

Collecting information about the professors' work and interests is one of the project's biggest problems. On the department websites, not all the data is available. Some professors might have their own personal webpages where they list their publications and areas of interest. In contrast, others might include it on their LinkedIn profile or on popular sites like Google Scholar and Academic Reasearch complete. It will be challenging for us to collect data regarding the professor's work from many websites. In order to retrieve the pertinent information about the professor and their work based on the keywords entered, the relevant information must first be gathered and stored in a structured format. This will make it easier for someone looking for someone with a similar academic interest to find the right candidate.

1.3 Research

1.3.1 Previous Research: Recommending Experts Using Knowledge Graphs

In a study conducted in 2019, researchers Behnam Rahdari and Peter Brusilovsky from the University of Pittsburgh addressed a problem similar to what this project aims to solve. Rahdari and Brusilovsky noticed that many undergraduate, master-level, and Ph.D. students struggled to find a research advisor who matched their interests and requirements [2]. Their struggles stemmed from perusing online sources that only provided a subset of information about different faculties' areas of expertise [2].

Moreover, there was a lack of an "expertise catalog" ---a catalog of expertise areas covered by faculty---resulting in students who were unable to properly identify their target area of interest or conduct an adequate search online for advisors [2]. To solve the problem, they developed an interactive search system built on a *knowledge graph*: a network of labeled nodes and edges that depict entities and relationships between entities, respectively [3].

They integrated data from Google Scholar and Wikipedia to help students find a research advisor or thesis committee member based on extracting several identifying features, such as scholar names, self-defined keywords from publications, and recent publications [2]. The primary data source was Google Scholar; data from Wikipedia added a semantic layer to profiles extracted from Google Scholar, providing stronger connections between keywords [2]. From this information, the knowledge graph was able to identify multiple connections between research topics and prospective research advisors within a large university or large research field [2]. Ultimately, the knowledge graph powered an exploratory search interface to recommend similar keywords and relevant scholars to students with limited knowledge and familiarity with a subject of research [2].

The interface consisted of four main parts: a search box, favorite keywords, favorite scholars, and recommendations. A user adds search terms, their preferred keywords, and their preferred scholars for the interface to display recommended keywords and recommended scholars. These recommendations are made using Cypher Query Language in Neo4j, the graph database management software used to build the knowledge graph [2]. The interface output is a list of the top 3 recommended keywords and the top 10 recommended scholars (see Figure 1) [2].

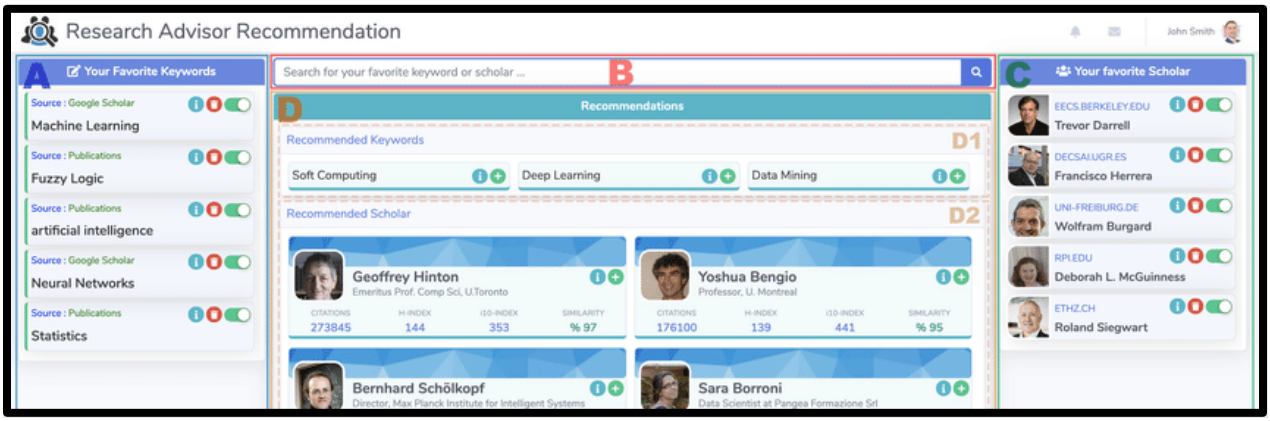


Figure 1. Front-end interface design for recommending research advisors. From Rahdari and Brusilovsky.

According to IBM, knowledge graphs use natural language processing (NLP) to construct a graph through a process called *semantic enrichment* [3]. The process allows knowledge graphs to understand relationships between different objects; this information is then compared to and integrated with other relevant and similar datasets [3]. Interestingly, the integration of multiple datasets within a knowledge graph can identify connections between entities that may not have been identified previously [3]. This project will rely on web scraping to extract data from online sources, NLP to search for relevant keywords and phrases, and a graphical user interface (GUI) application so a user can navigate the database with ease.

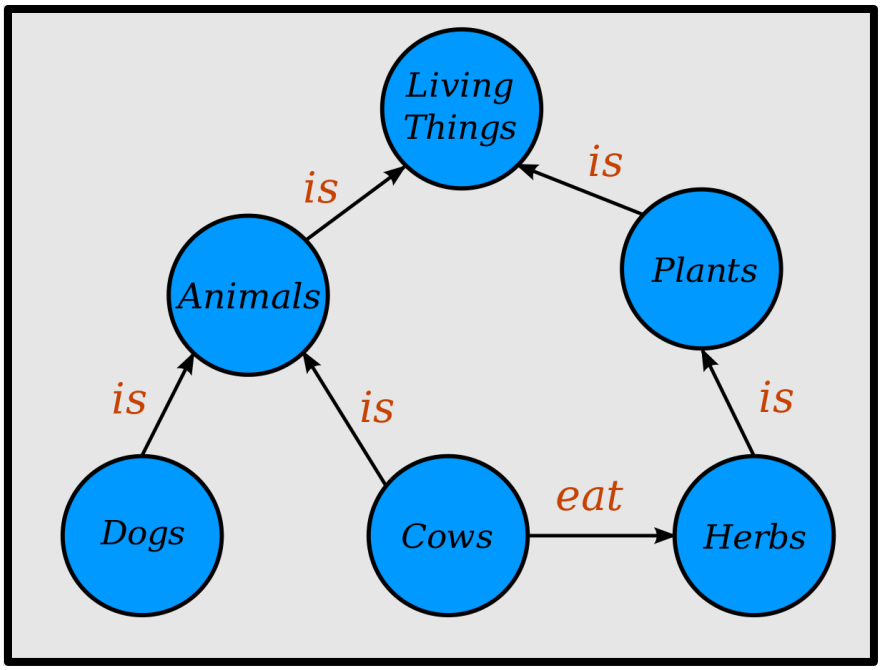


Figure 2. Knowledge Graph Example [4]

1.3.2 Web Scraping

The process of extracting data from a webpage is called web scraping. Usually, the collected raw information has to undergo a proper transformation before it can be saved in a useful format. In many cases, such as this project, manual web scraping is not practical merely because of the sheer volume of the data. Therefore, an automated approach will be implemented to achieve a faster and less demanding workflow. From a legal standpoint, it should be mentioned that web scraping is not illegal as long as the source of the data is available to the public. Therefore, this project will rely only on the data that exists in the public domain.

A typical web scraper is given a Uniform Resource Locator (URL) as the input. It then loads the corresponding webpage's HyperText Markup Language (HTML) code. HTML is a markup language that uses various types of tags as its primary mode of representation [5]. The scraper then parses the HTML code, extracts the pertinent data, and transforms it into the desired data format, like a CSV or JSON file. If the tags are sorted in a top-down Document Object Model (DOM) format, the appropriate data may be acquired by directly parsing DOM. Other general methods include vertical aggregation, XPath, and text pattern matching [5]. When it comes to selecting a web scraper, there are four aspects that have to be taken into consideration [6]:

1. Self-built vs. pre-built: There are many pre-built web scrapers available. However, when it comes to a specific project, it is better to use a self-built web scraper tailored to the project's needs, and the developer needs knowledge of computer programming.
2. Browser extension vs. software: browser extension web scrapers come in the form of add-ons that can be installed on popular web browser applications. They make the process of web scraping easy. However, when it comes to operations outside of the browser, they are not operable. In this case, software-based web scrapers are used.

3. User Interface (UI): A sophisticated UI goes a long way in helping the user extract all the pertinent information. It should be noted that if the UI is too complicated, working with the web scraper becomes frustrating.
4. Cloud vs. Local: In case of scraping big data or limited RAM/CPU capacity, it is recommended that the entire operation is done on the cloud.

Two major HTML tools in Python are "urllib" [7] and "requests" [8] libraries. The acquired HTML (or XML) codes can be parsed using libraries like BeautifulSoup [9] or Cheerio. Other available libraries for web scraping are Scrapy, Scrapy-Splash, JSpider, and PhantomJS [5].

In a previous similar study, areas of innovation in different companies in Germany were investigated by scraping their websites [10]. One of the research questions addressed in this study was "How do firm websites differ in terms of their size and content, and how does that interfere with web mining studies?". To address this research question, researchers developed ARGUS (Automated Robot for Generic Universal Scraping), a web scraper software based on the Scrapy library in Python.

1.3.3 Natural Language Processing (NLP)

Another method this project will use is natural language processing (NLP). According to IBM, NLP refers to the branch of artificial intelligence concerned with giving computers the ability to understand text and speech the same way humans can [11]. NLP combines rule-based modeling of language with statistical, machine learning, and deep learning models [11]. Some examples of NLP include voice-operated GPS systems, digital assistants (e.g., Amazon Alexa, Google Home, etc.), speech-to-text dictation software, customer service chatbots, and many more consumer conveniences [11]. NLP also plays a critical role in streamlining business operations, increasing employee productivity, and simplifying mission-critical business processes [11]. Among NLP's array of applications, this project will be focused on using NLP to process text-based data to identify commonalities across databases using keywords and phrases. This application of NLP is used in several machine learning applications.

One common example of NLP being used to identify commonalities in text data is spam detection. One can parse text in an email or a text message for language and patterns associated with spam. Indicators of patterns and/or words associated with spam include---but are not limited to---bad grammar, threatening language, spelling errors, and urgency. NLP enables the ability to identify patterns, words, and/or phrases that are attributed to spam. To make this determination, one uses a binary classification model: a model will learn from a labeled dataset of spam/not spam data to predict the likelihood of spam/not spam for an unlabeled data point. However, this project will not be using a machine learning model; instead, this project will be using the same principles of NLP as used passage similarity to identify similarities between two texts to classify faculty and researchers into areas of experience and expertise (*e.g.,* artificial intelligence, deep learning, cybersecurity, etc.). We used cosine similarity to measure the similarity between the articles and categories that they belong to by converting the text into vectors and then finding the angle between them. The values range from 0 to 1 where 0 refers to dissimilarity between the vectors and 1 indicates high similarity.

1.3.4 Graph Database

A customized, single-purpose platform for building and modifying graphs is referred to as a graph database. In a way that relational databases are unable to, graphs' nodes, edges, and attributes are

utilized to represent and store data. Data in the store may be immediately linked together due to the relationships, and in many situations, it can be accessed with only one action. The connections between the data are prioritized in graph databases. Relationships are permanently saved in the database, which makes querying them quick. Graph databases are advantageous for highly interconnected data since relationships may be easily displayed in them.

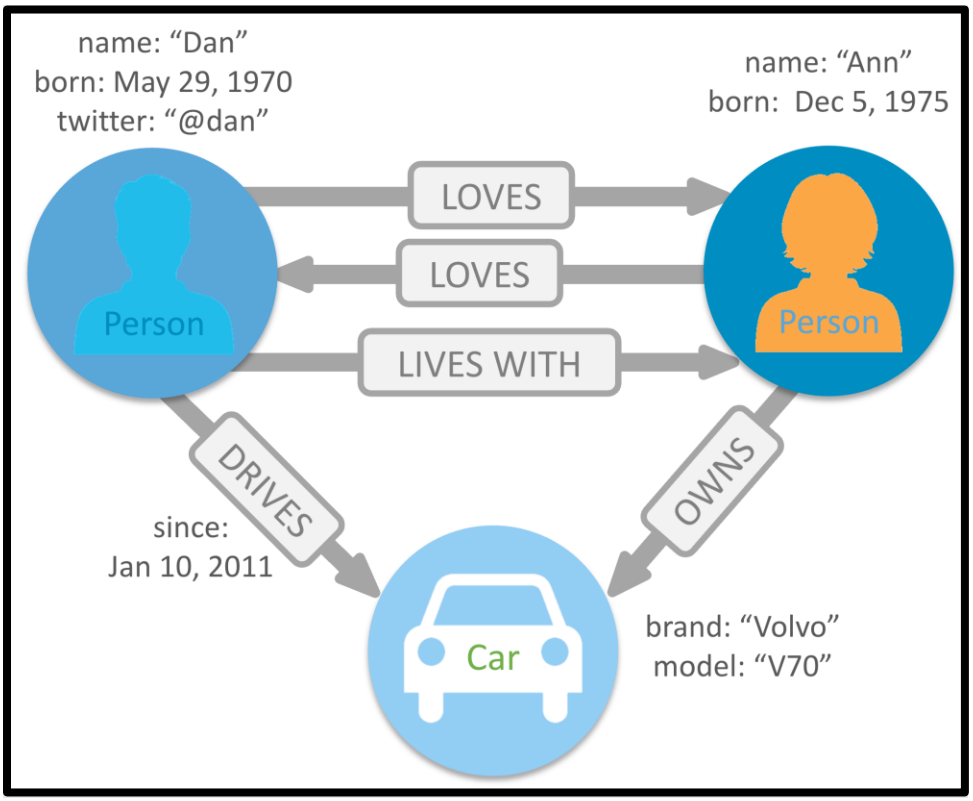


Figure 3. Graph Database example [12]

Instead of data points, the relational focus is on the columns of data tables. Both databases make it simple to add new data. A graph database's flexibility allows for the addition of new nodes and interactions between nodes, making it dependable for real-time data. Topographical data models are used by graph databases to store data. These databases join certain data points (nodes) and establish connections (edges) to construct graphs that the user may access using queries. Nodes can stand in for clients, businesses, or whatever information that a firm wishes to store. The database creates edges to help the user understand the connections between the nodes. Database administrators can scale large data quantities while still producing models that are useful. An RDF database, a sort of graph database that focuses on retrieving triples or data arranged in a subject-predicate-object connection, may be used by some enterprises. We chose Neo4j as our graph database management system. [13] [14] [15]

1.3.5 Neo4j

Neo4j is a leading software in the field of graph database management technology. Neo4j Inc. initially released it in 2007. Neo4j is helpful in showing and predicting the interrelations between items in a database. The core of this platform is the graph database. A graph database is suitable for graph algorithms such as link prediction, node embeddings, community detection, similarity, pathfinding,

centrality, and node classification. Neo4j, which according to its creators, is an ACID-compliant transactional database with native graph storage and processing, is offered in a closed-source commercial license for its online backup and high availability extensions and a non-open-source "community edition" that is licensed under a modified version of the GNU General Public License. [16]

The Neo4j graph platform is a set of applications whose purpose is to facilitate users' interaction with graph databases in various ways. Neo4j Bloom is a graphical interface that allows users to interact with the database in a visual environment without any prior programming knowledge. Cypher is a powerful query language inspired by SQL designed for graph databases. Cypher allows users to run multiple queries in a single transaction. Finally, Neo4j Aura is a cloud database service that allows users to run databases in the cloud without worrying about the infrastructure.

The Neo4j platform includes several connectors which facilitate working with existing architectures. Currently, connectors are available for Apache Spark, Apache Kafka, Business Intelligence, and Labs Integrations. In addition, Neo4j developer tools are designed to help users develop applications. These tools are Neo4j Desktop, which facilitates working with local databases; Neo4j browser, which lets users interact with databases in a browser; and Sandbox, which is designed to help develop new ideas.

Neo4j official drivers allow users to work with programming languages, such as C#, Java, Python, JavaScript, and Spring. The Neo4j community has also contributed to developing drivers for other programming languages such as R, PHP, and Ruby.

Many prominent companies such as IBM, Adobe, Walmart, COMCAST, Cisco, and HP use Neo4j to work with graph databases. In addition, there are nine successful case studies of Neo4j for real-time recommendation engines among more than 70 case studies listed on the Neo4j website [17].

1.3.6 NeoDash

NeoDash is an open-source tool for visualizing the Neo4j data. It enables dashboards, which gather visuals together and enable interactivity between reports. It allows the user to display the data in various ways, including tables, graphs, bar charts, line charts, maps, and more. It includes a Cypher editor so users may create the Cypher queries used to generate the reports themselves. It also allows the user to save the dashboards to the database and share them with others. It contains a Cypher editor to write the Cypher queries that populate the reports directly. NeoDash comes with built-in examples of dashboards and reports. It has the option to provide users access to a read-only, isolated dashboard after creating it. [18] [19].

1.4 Solution Space

Our team's solution to this project was to store data in a graph database with a user-friendly front-end platform that allows the user to search for researchers or topic experts of various subject fields employed by George Mason University's College of Engineering and Computing (GMU CEC). In order to produce this prototype, many steps were taken to obtain the data, clean the data, analyze the data, develop the graph database, and develop a front-end tool.

The first step of this process was to collect the data. In order to complete this step, we first needed to use web scraping techniques (done in Python with the libraries "requests" and "BeautifulSoup") to

collect all names of researchers and faculty members of the College of Engineering and Computing's (CEC) website. Once we collected the CEC's faculty list, we moved to the next phase of data collection, which was collecting information on what researchers and faculty are currently investigating.

This information was collected from various sources: their biography pages on the GMU CEC website, their LinkedIn profile's biography and "About me" sections, and from papers they have personally published or co-written. The abstracts from these papers gave us the paper's topic and, in hand, gave us the topic that the writer was researching at the time. In order to find these papers, we used web scraping on Google Scholar. Due to limitations with web scraping, we collected data from Academic Search Complete manually. Our primary focus for sources when collecting the research topics was on online research databases, namely Google Scholar and Academic Search Complete. It was brought to our attention that research and faculty personal pages might not be fully up to date with their current research.

Once the data were collected, it was cleaned and put in a uniform format before the natural language processing (NLP) analysis began. Another thing that was completed prior to the NLP portion was obtaining a list of keywords that could be research topics. This list helped our team identify the subject matter of the collected texts for each person in our expert list.

Once we collected the keywords, we began our NLP analysis. The method our team used to complete the NLP analysis was cosine similarity. This method compares the texts of two items and generates a score that measures the similarity between those two documents. The definitions of areas and categories within the dictionary were compared to the publications written by faculty members or the CEC website biographies the university writes for the faculty member. The publications and biographies were combined into one document for each faculty member. This new document was used to generate direct scores between the faculty members and areas and categories of the dictionary, while the scores generated for each publication was independent of those scores. These scores applied inferences of potential research interests for the faculty members. While these scores only infer potential research interests, data were also collected to establish definite links between faculty members and research topics. These definite links were the primary option for searching for research experts, and the cosine similarity scores were a secondary option if what the user is looking for could not be located with definite links.

Once we had a completed expert list, we then turned the list into a more digestible graph database. The graph database was developed using Neo4j, which applies the Cypher Query Language to create Java-based objects that store information. Relationships were created between those objects to allow the user to navigate through the database. With the database completed, we developed a front-end interface that allows users to find all experts on a given topic within the CEC. We considered a few potential options to develop a front end, including Microsoft Power BI, Tableau, Microsoft Azure, and Amazon Web Services. These platforms have cloud-based capabilities, meaning the user can access this prototype from any machine with proper server access. Every time the database is updated, the platform should automatically be given access to the new data. However, NeoDash was chosen as the front-end interface because of its ease of integrating with the graph database generated by Neo4j. It maintains the ability to automatically update the front end every time the database is updated because it is directly linked to the database. NeoDash is Neo4j's dashboard-based reporting tool, similar to

Tableau or Microsoft Power BI. Dashboards were developed to generate digestible information about the database and give the user the ability to search for a research expert.

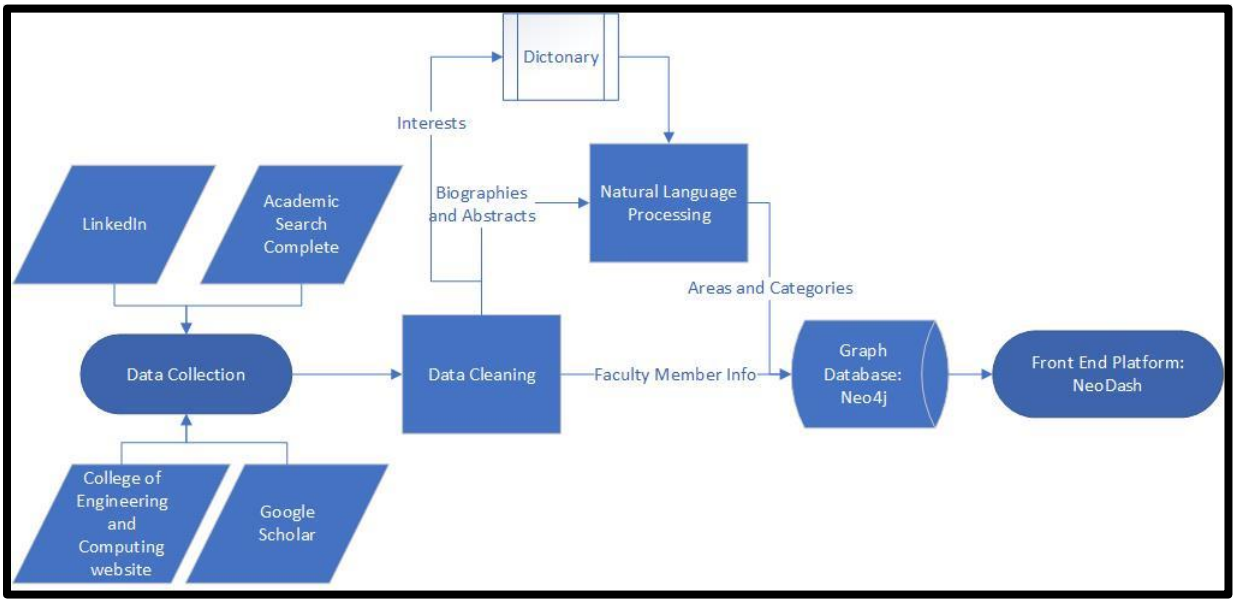


Figure 4. Solution Space Diagram

1.5 Project Objectives

Throughout the project, with the C4I & Cyber Center, the team worked on a variety of different aspects of data analytics, including data mining, NLP, systems engineering, and data engineering. After completing the project, we gained valuable experience in the previously stated knowledge domains through the necessary steps towards the project completion. For example, we gained data mining experience from web scraping the Internet to collect our data along with other potential data sources. In addition, we gained experience in NLP by extracting the research topics from the data we have collected. Finally, we also gained experience in systems engineering and data engineering when we developed the database and front-end product for the prototype.

Along with developing our data analytics skills, another concept our team learned was how to work with a team within the Agile development process. Agile development is a crucial concept in project management and development in the labor market. Any employee in the working environment must understand to provide value to one's team and enterprise.

The team's objective for an acceptable solution after completion of this project was a working prototype to provide the C4I & Cyber Center. In order to create a working prototype, it was vital for our team to develop a fundamental understanding of how research is conducted at GMU. This product should give the staff of the College of Engineering and Computing a device that makes finding staff or researchers on a given topic more easily determined and accessible. This product will simplify finding potential research partners for one's research and be a vital component for anyone conducting research within the College of Engineering and Computing. The product is an easily digestible front-end interface for users to easily manipulate the product to fit their needs. The team also left an easily understandable route to expanding if the client is pleased with the product. The client could then create an expansion

project to introduce the faculty and researchers and their research topics to the database for the other colleges at GMU to benefit from our product.

1.6 Primary User Stories

Our primary user stories covered the different components of the project we need to complete to provide a meaningful tool. These primary user stories encapsulate the solution space:

- **Obtain data –** "As a developer, I would like to find out which organizations GMU faculty and researchers collaborate with so I can quickly identify data sources with precision"
- **Apply NLP to data –** "As a developer, I want to extract and categorize data so I can accomplish one of the primary goals of this project"
- **Store data in some database –** "As a developer, I want to store data in a graph or SQL database so I can use a reliable database to store the data we extract and make connections in our dataset"
- **Develop GUI –** "As a developer, I want my client to have a working GUI so the client can use our tool without having to run or interpret any code"

1.7 Product Vision - Sample Scenarios (why would someone want to use this)

Scenario #1

This product enables the researchers to find suitable research partners when submitting a grant or project proposal. This product gives the ability to find the academics of various departments of the University and their pertinent work to make it simple for someone looking for a subject matter expert to find the right individual. This product will be used by researchers to find the relevant subject matter expert by simply searching with keywords instead of gathering the information by visiting numerous websites, which is a tedious and time-consuming process. In addition, this product enables the researcher to quickly identify the academics working on similar domains and collaborate with them to conduct research.

Scenario #2

This product can be used by the administrative staff of GMU. The administrative staff has been constantly approached by outside companies or researchers from other schools looking for relevant subject matter experts to collaborate with or consult. It will be of significant use if the administrative staff have all the information about their academics and their ongoing research, which can be used to guide the companies or researchers from other schools seeking to find the right individual. This product enables the administrative staff to keep track of all the research done by various professors on a wide range of topics.

Scenario #3

The students who intend to pursue a Ph.D. will find this product useful. Finding a professor who is working on a project that interests the student is the first step in pursuing a Ph.D. Finding the right professor to complete his Ph.D. requires the student to laboriously compile all the pertinent information on the academics from numerous sources. Our product enables the students to find a

professor just with keywords. Students can quickly locate pertinent subject area experts with the aid of this project and contact them with a proposal to pursue a Ph.D. under their guidance.

1.8 Definition of Terms

AFCEA – The Armed Forces Communication and Electronics Association is a nonprofit membership association serving the military, government, industry, and academia as a forum for advancing professional knowledge and relationships in the fields of communications, information technology, intelligence and global security.

Cyber Security - Is the protection of computer systems and networks from information disclosure, theft of, or damage to their hardware, software, or electronic data, as well as from the disruption or misdirection of the services they provide.

C4 architectures – Also known as C4 model, is a lean graphical notation technique for modelling the architecture of software systems.

ICRRTS - International Conference on Refrigerated Road Transport and Safety aims to bring together leading academic scientists, researchers and research scholars to exchange and share their experiences and research results on all aspects of Refrigerated Road Transport and Safety.

Information Technology (IT) - Is the use of any computers, storage, networking and other physical devices, infrastructure and processes to create, process, store, secure and exchange all forms of electronic data.

Interoperability - The ability of computer systems or software to exchange and make use of information.

Multi-modeling - Is an approach to improve efficiency of adaptive system or observer system.

MySQL - Is an open-source relational database management system.

National Institute of Standards and Technology (NIST) - Is a physical sciences laboratory and non-regulatory agency of the United States Department of Commerce. Its mission is to promote American innovation and industrial competitiveness.

National Science Foundation (NSF) - Is an independent agency of the United States government that supports fundamental research and education in all the non-medical fields of science and engineering.

NATO - The North Atlantic Treaty Organization, also called the North Atlantic Alliance, is an intergovernmental military alliance between 30 member states – 28 European and two North American.

Neo4j - Neo4j is a graph database management system developed by Neo4j, Inc.

Oracle - Is an American multinational computer technology corporation that sells database software and technology, cloud engineered systems, and enterprise software products.

Structured Query Language (SQL) - Is a standardized programming language that is used to manage relational databases and perform various operations on the data in them.

STIDS - Semantic Technologies in Intelligence, Defense, and Security is a premier opportunity for collaboration and cross-fertilization between researchers and practitioners of semantic-based technology with particular experience in the problems facing the Intelligence, Defense, and Security communities.

2 Data Acquisition

2.1 Overview

The data are extracted from a variety of websites, namely the George Mason University College of Engineering and Computing (GMU CEC) website, LinkedIn, Google Scholar, and Academic Search Complete. We created a database of professors and the research they have performed. The database included professor name, email, contact number, areas of research, and subtopics within those areas of research. While performing the web scraping, we added a few additional attributes to the database.

The GMU CEC website has information about the faculty, and the data source type is open source. With the help of the faculty information, we can easily find a professor interested in a specific area of research; getting in touch with the professor to work together or consult on a certain study subject would be simple. Information extracted from the GMU CEC website was used to search for more data in other data sources, like Google Scholar.

Name, title, fields of study, citation count, and a profile link were just a handful of the details available on Google Scholar. This list was adjusted against the list of names acquired from the GMU CEC website because it also contains any profile verified by a GMU email address, which included retired professors, graduate students, and our target group. Figure 5 shows a screenshot of the format of a Google Scholar list of GMU people. These profile links and names were then used to collect the publications for each name in the adjusted list. Each researcher profile listed their publications, sorted by the number of citations. Figure 6 shows an example of a Google Scholar profile. Furthermore, in order to assist researchers in locating professors who are subject matter experts, Google Scholar information was used to extract keywords. More information was collected on professors' LinkedIn pages.

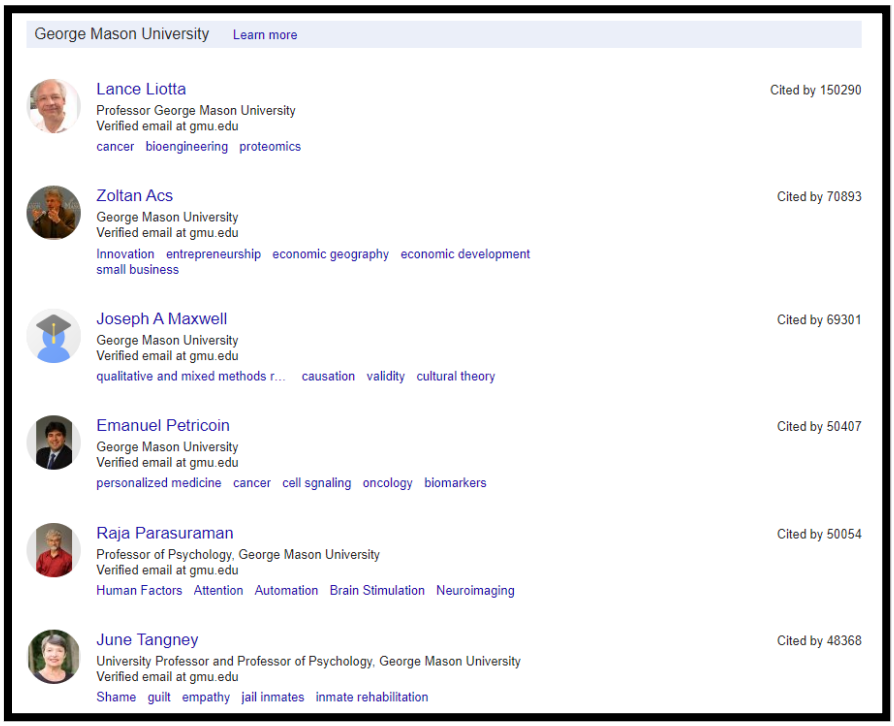


Figure 5. Six highly cited Google Scholar profiles affiliated with GMU

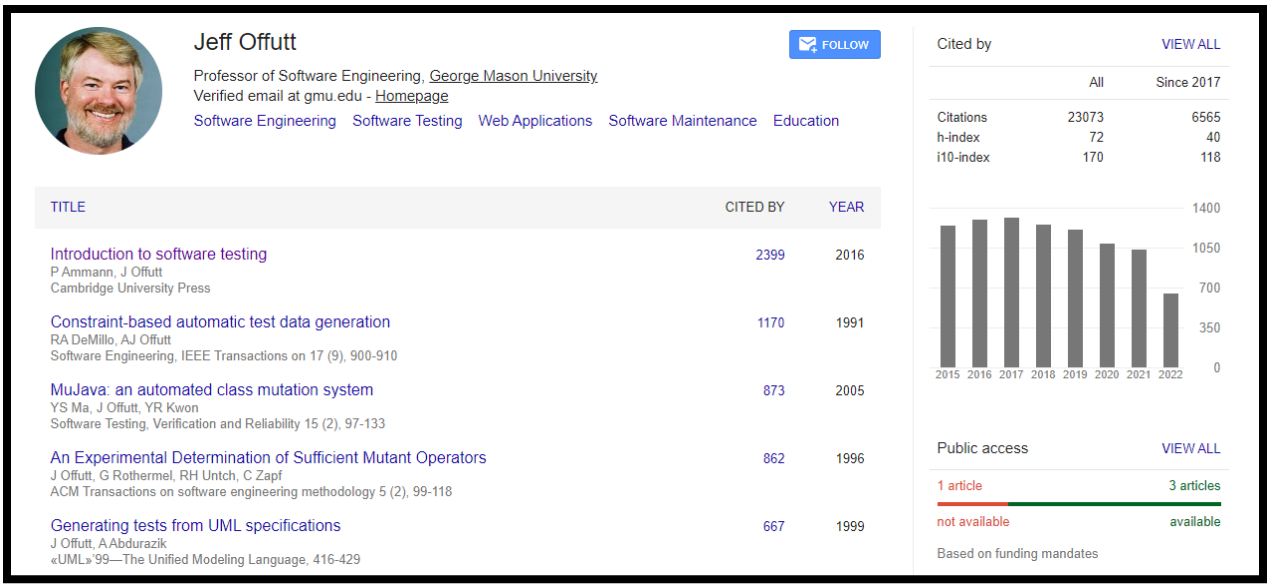


Figure 6. Sample Google Scholar Profile

LinkedIn has data related to the research performed by a professor and their listed publications. The about sections of faculty members' profiles can contain information about their expertise or research interests. The documents that have been published on LinkedIn are documented by publication date, title, and description. These publications are treated the same way google scholar documents are treated. Because academics do not often update their LinkedIn pages, we cannot ensure that the information is accurate. Finally, data from Academic Search Complete was used to populate the database.

Academic Search Complete is a scholarly, inter-disciplinary full-text database designed for academic institutions. It consists of thousands of periodicals, peer-reviewed journals, indexing, abstracts, and several other pieces of literature, providing a vast amount of research available with a quick search online. The database has content from as early as 1887 in PDF format; most of the content available in Academic Search Complete is saved in native (i.e., searchable) PDF format. Users can conduct searches based on a variety of "searchable tags," including author, author affiliations (to search for organizations and/or institutions affiliated with authors), date of publication, subjects, title, and journal name. It should be noted that access to Academic Search Complete is provided through an institution login. GMU has access to the database, but one must use Multi-Factor Authentication (MFA) to access the database with a GMU account. To extract the information from the search queries, Elton Bryson Stephens Company (EBSCO) offers the option to export search results to a file. Results were exported in a brute force method, where each file was generated manually.

All acquired data were consolidated in a CSV file, which was used to populate the graph database.

2.2 Field Descriptions

The field descriptions in this project were broken up into the variables that were used in the graph database and variables that contribute to the collection of data used in the graph database. The following fields made up the contents of the graph database:

- Professor or researcher name (Type: string) – The names of the GMU-affiliated faculty and researchers. These names are pulled from the faculty listed on the CEC website and all external websites.
- Affiliated areas and categories (Type: string) – Areas and categories are terms designated by faculty members themselves. An example of an area and a category is "computer science" and "artificial intelligence," respectively. It is expected that if a faculty member at GMU has an area of expertise, it would be noted either on the CEC website or on an external website.
- Phone number (Type: string) – Phone numbers extracted from external websites. The phone numbers will be stored without any punctuation.
- Email address (Type: string) – Email addresses extracted from websites.
- Degrees (Type: string) – The degrees conferred to faculty or researchers. Extracting degrees populates the list of topics and areas of expertise for a faculty member or researcher.
- Title of publication (Type: string) – The title of a journal article. This information is used to reference across multiple websites to ensure duplicate data are not being extracted.
- Publication date (Type: string) – The publication date of journal articles, scraped from the automated web scrapers. The dates are stored in YYYY-MM-DD format.

The following made up other variables collected. These variables provided information for populating the database with keywords. Some of these fields may be null. These variables contribute to the collection of data used in the graph database:

- Text of abstracts (Type: string) – The block of text from the abstract of a GMU-affiliated faculty member's publication. This block of text is used to provide additional keywords.
- Faculty member biographies (Type: string) – The block of text from the biography of a GMU-affiliated faculty member. This information is scraped from the CEC website or external website, such as LinkedIn or Google Scholar. Biographies are important because, like the text of abstracts, keywords and topics are extracted from the block of text.

2.3 Data Context

2.3.1 George Mason University College of Engineering and Computing Website

The information about the GMU faculty was taken from the GMU CEC website. This dataset contained information on the faculty members, including their names, contact information---such as phone number, email, title, biography, research interests, and degrees---and their research titles. This faculty information made it simple for us to locate the professor with a particular research interest. Furthermore, since all the professors' contact information are available online, getting in touch with the professor to collaborate or consult on a particular area of research would hypothetically be easy. The scraped data were stored in a CSV file which was imported into the database.

2.3.2 LinkedIn

The information collected from the LinkedIn website consisted of details related to a professor and his or her research. This dataset contained information on faculty members, such as topics, titles, subjects and keywords of research, and biographies. The faculty members listed their research areas and publications on their LinkedIn profiles, which were used to generate keywords for research topics. These keywords were generated from the abstract of listed publications using NLP. This dataset had some missing data because not all faculty members had a LinkedIn profile. The scraped data were stored in a CSV file and was imported into the database.

2.3.3 Google Scholar

The dataset retrieved from Google Scholar consisted of the article title, authors, publication date, URL, and description. In addition, this dataset offered details such as title and abstract, which were used to generate the keywords needed to search for a subject matter expert. Figure 7 shows an example of data that is available for each publication on Google Scholar. This information was useful in associated keywords related to a particular researcher. These keywords helped with finding professors that were experts in a particular topic. This dataset was also retrieved through web scraping, and the results were stored in a CSV file, which was imported into the database.

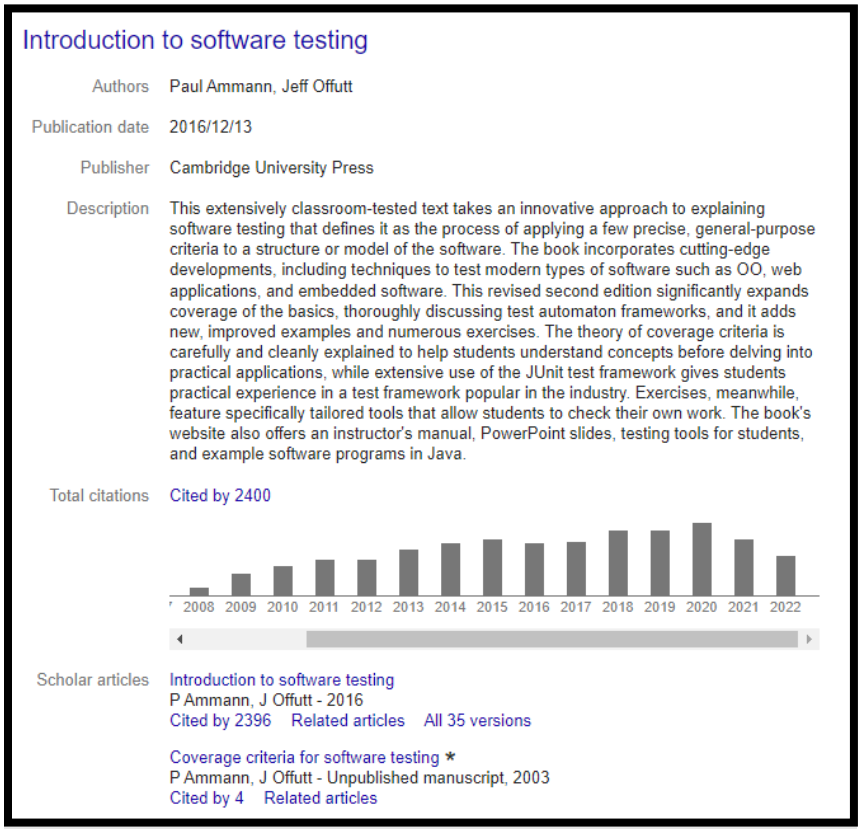


Figure 7. Available publication data on Google Scholar

2.3.4 Academic Search Complete

The dataset gathered from Academic Search Complete comprised of article title, authors, publication date, subjects, and keywords. This dataset provided the details about the professor's name and the title of the article that they authored, which were used for recommending the subject matter expert when searched based on a research topic. In addition, this dataset also provided additional information about each publication, such as the publication date and keywords. All this data was collected manually from the Academic Search Complete website and stored in a CSV file to be imported to the graph database.

2.4 Data Conditioning

Due to the nature of web scraping, data completeness and quality were determined entirely by the amount of data our team could collect from the Internet. Each source required the development of a new web scraper in order for data to be collected; in the case of Academic Search Complete, the data collection was done manually. This section describes the processes conducted for data collection and conditioning from each source.

2.4.1 George Mason University College of Engineering and Computing Website

The data from the GMU CEC website was collected using BeautifulSoup and Selenium packages: Python packages that play a role in web scraping. When trying to collect the data using the Requests package to parse the HTML code for the webpage, access was blocked. In order to bypass this blocker, Selenium was used to open a new instance of a Google Chrome driver to navigate to the webpage. Once the driver reached the "Meet Our Faculty" page, the name and URL of a person's profile were collected for each name in the faculty list. The driver then went to every person's university profile page to collect their phone number, email address, title, biography, research interests, and degrees. The website is constantly being updated, so some faculty profiles are not complete. For example, some people are missing one or nearly all of the previously stated collected data. There are even some people missing entirely from the list.

2.4.2 LinkedIn

The data from the LinkedIn profiles of members of the CEC required a web scraper to be developed for collecting the data within the project's scope. The BeautifulSoup and Selenium packages were used in the web scraper. Before collecting data from LinkedIn, a URL was needed for each member of the CEC that has a LinkedIn profile. To obtain the list of URLs, our team relied on Google search results. The phrase "[Person Name] George Mason" was entered into each LinkedIn search, and the web scraper took the first result.

Once the LinkedIn profile URLs were collected, a new Google Chrome driver was opened and controlled by Selenium and navigated to the LinkedIn login page. First, it logged in to LinkedIn to gain access to user profiles. Once the driver was logged in, it navigated to every user profile and collected the contents in the "About" section. Along with this, the scraper attempted to go to the publications section of the profile using a different URL. Anyone with publications on their profile was found with the URL "[profile_URL]/details/publications". This URL contained every publication the owner has added to their profile. The title, publishing date, and description were collected and added to a CSV file along with the name of the profile's owner.

While the scraper was collecting the "About" sections and the publications of each profile owner, LinkedIn's bot detection software detected the web scraper and temporarily blocked the LinkedIn account the scraper was using. Unfortunately, the scraper had only gone through 70% of the names from the CEC list. Therefore, the last 30% of the data had to be collected manually. The manual collection process was completed by copying and pasting the "About" section and the title, publishing date, and description, if there was one, into the CSV file that the web scraper had created. In the future, any other names that are added to the database and need to have data collected on them will either have to be collected manually, or the account the scraper uses to access LinkedIn will have to be switched to ensure no account gets permanently banned.

2.4.3 Google Scholar

The web scraper for Google Scholar was developed in Python using the Selenium package for downloading web content and performing operations like clicking buttons and links. The downloaded content was then processed using the BeautifulSoup package. Using Google Scholar as a data source assumed that each active researcher had a unique profile in Google Scholar. Publications are added automatically to people's profiles by Google.

First, a list of all profiles validated by a GMU email address was scraped to create a dataset from this source. This list was presented in groups of 10 on each page, sorted by the number of citations. At this step, a list of about 1,700 profiles was created with each record, including name, title, fields of study, number of citations, and profile link. In addition to our target population, this list included retired professors and graduate students, so it was refined against the list of names obtained from the GMU CEC website. Each profile in Google Scholar presented the publications in a list of 20 and added 100 more publications each time the "show more" button was clicked. Each publication in a profile was linked to another page from which the title, abstract, and journal/conference/book URL was scraped.

2.4.4 Academic Search Complete

The data from Academic Search Complete was collected manually due to the design of the online database. There were a variety of issues that would have to be tackled in order to make a web scraper for the database. The first issue was that it required the user to log in to their GMU account before entering the database, and that login required two-factor authentication. Due to the two-factor authentication, automating this process was taken entirely off the table. After logging into a GMU account, one must renavigate to the database. If a person tried to access the database through a URL, the website redirected them to another webpage. The final issue occurred once reaching the database. While adjusting the URL changed the search results of the database, the results stayed the same. The only way to navigate the database is to type in the article or author being searched manually. These issues could be dealt with in the future, but due to the time constraint, the data from Academic Search Complete were collected manually.

Before exporting search results, search terms had to include an "author affiliation"—recognized by Academic Search Complete with the tag "AF"—which tells Academic Search Complete to search for articles affiliated with a particular institution or organization. This became especially important because some faculty had a name so common that a search for their name alone returned over 4,000 articles. For example, the query *AU "Wang, Yue"* returned 4,384 results, but the query *AU "Wang, Yue" AND AF*

"George Mason University" returned 5 results. Without the author affiliation field, extracting relevant data from Academic Search Complete would be very difficult.

To collect the data, our team used the folder function provided in the database. This function allows one to save all search results to the folder and later download the folder as a CSV file. The CSV file provided all information available on each document's webpage. The necessary information that was taken from the CSV file were Article Title, Authors, Publication Date, Subjects, Keywords, and Abstract. The rest of the information provided was irrelevant to this project's scope.

Before conducting analysis, one of the goals with the Academic Search Complete was to keep GMU-affiliated entities only within the "Authors" field. Academic Search Complete exports search results with all contributors in an article. For the purposes of this project, only contributors affiliated with GMU were of interest. To extract this information, each author in the Academic Search Complete dataset was isolated from one another, but retained the same surrounding information (e.g., if a journal article had Authors A and B as collaborators, their entries were now stored as a row of data for Author A and a row of the same data for Author B). Then, the ordering of names in Academic Search Complete had to be switched from "'Last name', 'First name'" to "'First name' 'Last name'" because to identify contributors affiliated with GMU, their strings were matched according to the list of names collected from the GMU CEC website, which stored names in "'First name 'Last name'" format. Additionally, names had to be isolated because some journal articles had multiple contributors from GMU; to store this information in the graph database, each name had to be considered separately. Below is an overall example of the data transformation in the "Authors" field that had to be done to isolate Dr. Leonard Adelman, professor of Systems Engineering and Operations Research in the CEC:

```
Adelman, Leonard; Lehner, Paul B.; Cheikes, Brant A.; Taylor, Mark F.
                                    ↓
                              Leonard Adelman
```

2.5 Data Quality Assessment

2.5.1 George Mason University College of Engineering and Computing Website

- **Volume**: The amount of data we scraped from the GMU CEC website was based on the number of researchers and faculty that the CEC website had listed on its faculty page. These people were the only ones on whom data were collected because the names of adjunct researchers and faculty were not readily available on the website. There were approximately 235 faculty members in the CEC at the time of conducting this project. Considering this, the amount of data collected from the CEC website was approximately 330 KB. This volume of data collected can change based on the amount of information submitted by the faculty member to the school and on what the faculty member thinks is relevant. This information can cause the size of the biography, research, and research interests to change size.

- **Variety**: The data collected from the CEC website were various types. Most of the data collected were text information like Biography, Research, Research Interests, Degrees, and Job Title. These fields were used in the analysis part of the project. In contrast, fields like email, phone number, location, and name were added to the complete data set since these fields were not

necessary for analysis. Another piece of information that was collected were URLs for personal websites that had relevant information.

- **Velocity**: The velocity of the data was based primarily on two factors: the frequency at which the website was updated and the frequency at which we ran the web scraper to collect the data. The CEC has the possibility of gaining or losing faculty and researchers every semester. Because of this, web developers need to update the website to add new faculty member profiles constantly. The issue with this process is that there is not a deadline for profile addition. Since there is not a deadline, our team was unsure when the new information would be added, and because of this, we are unsure of when to rerun our web scraper. Our team could run the scraper once at the end of every semester, run it once during midterms, and once during finals, or run it every day. It depends on when and how frequently the website is updated and how pressing it is to have the new profile information.

- **Veracity**: When it comes to the veracity of the data, it is readily available since it is on a webpage that does not require a password to gain access. When it comes to the quality of the data, that is a different story. The data is highly accurate, considering how it is collected and added to the webpage. Web developers receive the information from their supervisors, who receive it directly from the faculty (or even information the faculty have written themselves). Regarding the conformity and completeness of the data, our team received some information from the web developers about broken links to faculty personal websites, which created a lack of data for some faculty. Even when links work correctly, not all of the data items that we planned to collect were there. Some profiles were missing email, phone number, or research interests. Some profiles were even missing all of these items.

  Along with that, some faculty profiles provided much more information than others. For example, while some profiles contained a research history section, a research interest section, and a personal website link, others contained none of those. The only commonality between all profiles was that they all contained a biography section, a contact information section, and a degree section. Even with these commonalities, some faculty members chose to put more thought and go further in-depth during their biography section. One good thing that the profile pages had was their lack of uniqueness. Every profile had the same format, and even though some sections were missing from the profile, the page format was the same for every member. The format made web scraping the CEC webpage relatively simple.

2.5.2 LinkedIn

- **Volume**: The amount of data available on LinkedIn for our team was based entirely on the amount of effort faculty members of the CEC put into their individual LinkedIn pages. Some faculty members did not have LinkedIn profiles due to the length of time they had been employed at the University. Some people added all of their relevant information like Employment, Licenses & Certifications, Education, and Publications. While on the other hand, some added the bare minimum, like one sentence “About” sections. Another factor that dealt with the volume was the amount of research and writing they conduct and the amount to choose to add to their LinkedIn profiles. Some chose to add every one of their publications to

their profile along with a description, and others added none. After scraping LinkedIn profiles, we identified 90 “About” sections and 375 publications.

- **Variety**: The types of data collected from LinkedIn are the “About” and “Publication” sections. If a faculty member had an in-depth profile, the “About” section was most likely where they would talk about their research interests. The publications provided our team with definitive proof of what they have researched in the past and possibly what they are currently researching. The other sections on a LinkedIn profile either did not fall under the scope of the project or did not provide enough information to be valuable to the project.

- **Velocity**: The velocity of the data was based on two things: the rate at which the CEC faculty page is updated and the frequency at which individual LinkedIn staff profiles were updated. LinkedIn profiles are controlled by the profile owner, so it is unknown how frequently the profiles are updated. The web developer updates the faculty page and, as stated previously, there is no hard timeline on when the page is updated. The web developers hope to fully update the faculty page by the end of the semester, but that is not guaranteed. The only way for our team to collect data from LinkedIn was to obtain the names from the CEC website. The LinkedIn data collection was entirely dependent on the data collection from the CEC, so it should be run every time new names are detected within the data.

- **Veracity**: When it comes to data availability, LinkedIn has many profile settings that can change how viewable people’s profiles are. Some profiles were entirely seen by anyone, even if the viewer did not have an account. Some accounts had restricted settings where only people with LinkedIn profiles could see the profile. Finally, some profiles could be hidden entirely so no one could see them. Our team’s best option for web scraping LinkedIn data was to log in with someone’s account so we could see as many available profiles as possible. The alternative option was manually collecting the data if our team's access was revoked. This process is much slower than web scraping, so this option was our last resort. Nothing could be done about the private profiles except for emailing the profile owner, at which point the profile would no longer be needed.

  The data obtained from LinkedIn was not fully complete because not every faculty member has a LinkedIn profile, and profiles could be private. Each profile had some factor of uniqueness since every section did not need to be included in every profile. For example, every profile requires an “About” section but does not require the “Publications” Section. LinkedIn profiles had three things going for them: accuracy, atomicity, and conformity. It can be concluded that the profiles had a high degree of accuracy since the owner controls the profile. Because of this, we knew there was no false information in the “Publications” section. However, there could be missing publications if the profile has not been updated recently. There was a high level of atomicity because our team could pick and choose what profile sections we need when web scraping or through manual collection. We did not need to retrieve the entire page for this project; just the relevant sections and the profiles gave our team the ability to do that. Finally, there was a high level of conformity because all LinkedIn profiles have the same webpage format. Even if some sections were missing, the actions taken during web scraping were the same for each profile.

2.5.3 Google Scholar

Google Scholar is a free search engine used for indexing scientific literature in terms of both full text and metadata. Google Scholar was released in 2004 with the motto "stand on the shoulders of titans" [20], and has gone through many changes over the years. Google Scholar search results include academic journal papers, conference papers, books, dissertations, technical reports, abstracts, and patents [21]. This platform uses a web crawler to return search results, covering an estimated 100 million articles written in English [22]. Compared to similar search engines, the search options are limited; however, Google Scholar lets researchers create profiles using their institutional email addresses to list their publications and citations. There is no option to search people or articles by the institute. However, if a list of researchers is already available from another source, the list of publications and citations can be retrieved from Google Scholar. Compared to other search engines like SCOPUS and the GMU library, the publication list of researchers seems to be more comprehensive. Searching for the keyword "@gmu.edu" showed that since the year 2000, about 19,400 articles have been published with at least one author affiliated with GMU at the time of publication. This time range was selected according to client instructions. The data features are discussed below.

- **Volume**: For each publication, several items, including title, authors name(s), journal, publication date, abstracts, and keywords, were stored. Since the data was in text format, considering 2kB for each record seemed reasonable. The total storage volume needed for this dataset was about 40 MB.
- **Variety**: The data to be collected from this data source was in text format. The title, author(s), date, and journal name are all structured data. Keywords are semi-structured data because the number of keywords varies from one article to another. Also, some data sources use different words to identify keywords like "subjects" in the case of "arxiv.org". Abstracts are plain text data and can be considered unstructured text data.
- **Velocity**: Publishing scientific articles is not an easy undertaking and, in most cases, is very time-consuming. Given that and the limited number of researchers in the GMU community, it seems reasonable to update the dataset from Google Scholar in a month.
- **Veracity**: Google Scholar is not a dataset; it is a search engine that indexes articles on their corresponding websites. With that in mind, the level of accuracy gained from this data source was equal to that of the publication web pages. Regarding accessibility, Google Scholar data is publicly available, and no account login is required. However, sending too many requests from an IP address will cause some form of a ban. If requests are coming from a personal IP address, they would be banned for a few hours, and if they come from an academic IP address, there would be a partial ban, including a CAPTCHA test. Almost 1,700 profiles on Google Scholar were verified with a GMU email address. People can use any name to create their Google Scholar profiles; hence there are many cases in which people used nicknames, middle names, or initials. There were 107 exact matches with the list that was acquired from the CEC website. The contents of each profile included the list of publications updated automatically by Google, and it is one of the most accurate sources of information about this matter. In terms of atomicity and conformity, all profiles have the same structure, with each publication linked to another Google Scholar page listing title, authors, date, and description of the publication, along with a link to the external publication webpage.

2.5.4 Academic Search Complete

- **Volume**: Academic Search Complete has more than 8,500 full-text periodicals, including more than 7,300 peer-reviewed journals, indexing, and abstracts for more than 12,500 journals and a total of more than 13,200 publications including monographs, reports, conference proceedings, etc. Searchable cited references are provided for more than 1,400 journals. The number of options available through Academic Search Complete made it a reliable data source. In the context of this project, we identified 746 instances of at least one GMU faculty member affiliated with a research article in Academic Search Complete. Some articles consisted of more than one GMU faculty member; those instances were counted once per GMU-affiliated author for our dataset.

- **Variety**: Academic Search Complete has literature that spans dozens of fields of study. Academic Search Complete offers full-text coverage in dozens of areas of academic study, including animal science, anthropology, area studies, astronomy, biology, chemistry, civil engineering, electrical engineering, ethnic & multicultural studies, food science & technology, general science, geography, geology, law, materials science, mathematics, mechanical engineering, music, pharmaceutical sciences, physics, psychology, religion & theology, veterinary science, women's studies, zoology, and many other fields. This project used entirely text data from several fields, such as Author, Keywords, Subjects, and Abstract.

- **Velocity**: The velocity of the data is based on how many entries the Academic Search Complete pulls. According to sources, Academic Search Complete is updated daily. It is unclear how many additional sources can be queried each day.

- **Veracity**: Academic Search Complete has a reputation for being a comprehensive database of tens of thousands of journals, consisting of articles that have gone through a strict and thorough peer review process; the overall quality, completeness, and accuracy of the data is high. The articles that can be queued in the Academic Search Complete database are scholarly articles. In terms of atomicity, searches either return results or no results at all. With respect to conformity, the database aligns with industry-standard search capabilities: users are able to search based on different tags, such as author name, author affiliation, ISBN, and many more. It is considered to be one of many metadata services, but what makes Academic Search Complete unique from other services is its subscription-based membership and its rigorous curation and indexing of open-access journals.

2.5.5 GMU Library Portal

- **Volume**: The amount of data we received from the University Library's website was based on the number of researchers and faculty from the CEC department who publish their articles. Almost all the articles can be accessed through the GMU Libraries. The search results of the professor's name returned the articles associated with the professor and other materials, such as books and videos which were not considered for this project. All the articles published by the professors are available on Mason libraries and can be accessed with a GMU account.

- **Variety**: The data collected from the Mason Library represented a wide variety of academic fields ranging from anthropology to zoology. These fields depend on the researcher's interest and work in the respective field. The articles gave our team concrete evidence of their

research's veracity. The Mason Library also provided the details of the subjects that the professor is a part of, which can be used to find the articles that were published by the academic in that field.

- **Velocity**: The velocity of the data is based primarily on two factors, the frequency the website is updated and the frequency of the articles published by the researchers. The Mason Library constantly needs to update its website with new research articles that are published. However, the frequency of the updating process is unknown.

- **Veracity**: The data's veracity depends on the veracity of the journal articles it pulls. Most of the articles are peer-reviewed which provides authenticity to those articles. Mason Library has rules stating that the articles should go through a strict and thorough peer review process to ensure high data quality. Regarding the completeness of the data, some articles have missing fields like the description, which creates a lack of data for some articles. In addition, there is a high level of atomicity since, when using online scraping or manual collecting, our team may pick and choose which profile sections we require. The data is highly accurate, considering the articles are scraped from trusted databases. However, some professors have the same first and last names, which created a problem while searching for articles published by a specific professor. The portal does not provide a way to filter query results based on author affiliation. This issue puts the value of this data source into question.

2.6 Other Data Sources

2.6.1 NIST and NSF

NIST and NSF were considered as two auxiliary data sources. The NIST dataset includes about 40,000 publications between 2000 and 2022. Each record in the NIST dataset includes the title, publication date, author names, abstract, field, keywords, and a URL to the publication. This dataset could be used to create an exhaustive list of the keywords which can be used to train NLP models. NSF is a much broader dataset that includes grant documentation in all the fields of science and technology. The advanced search on the NSF website makes it possible to search the dataset according to the date of publication, the field of research, state, institution, etc. The sheer number of records available in this dataset makes it impossible to acquire the entire dataset in a single inquiry. One has to break down the dataset into multiple inquiries to store all the records, making the web scraping process more complicated. These datasets can be used to cross-reference names and research interests of the GMU researchers.

2.6.2 Taxonomy

Considering that all of the data collected from our sources were analyzed using NLP, our team developed a well-defined dictionary. A dictionary was needed to categorize the CEC professors and researchers into their fields of research and identify the subjects of articles and how they relate to research fields. The basis of this dictionary was the Volgenau School of Engineering (VSE) Research Portfolio Taxonomy. This document contained the subject areas everyone in the CEC is currently researching. The information is split into three sections: Topic, which is a larger field of study; Area, which is the specific subject with the topic; and Category, which is more specific domains within the

area. For example, the topic of “Digital Systems and Data” has an area of “Autonomous Systems,” and the different categories within that area are “Robotics” and “Unmanned and Autonomous Vehicles.”

The Taxonomy also contained a separate page for each topic. Each page contained the people and departments conducting research in certain areas and categories. The page also contained whether or not a person has received funding for their research of over $100,000 and who is providing the money. The page also contained up to 3 publications the person has published about the related category. These pages for each topic were another great source of information in classifying people in specific fields because it was already partially completed for our team.

While the Taxonomy only contained information on grants received that provide over $100,000, the CEC posts any grants it wins, no matter the amount rewarded. That information can be scraped from the official CEC website. The information collected consisted of the name of the organization rewarding the grant, the name of the grant, the name of the grant winner, the award amount, and the date awarded. This data provided keywords for the dictionary and further information on people of the CEC conducting research.

2.6.3 Removed Sources

Other data sources considered for this project included Research Gate and Arxiv.org. These two sources are online research databases that store articles and documents similar to Google Scholar and Academic Search Complete. After conducting a quality assurance analysis on these two sources, we determined that these sources would be unnecessary for this project because nearly all of the articles stored in these databases are also included in Google Scholar, Academic Search Complete, or we can gain access to them through the GMU Library Portal. Collecting the abstract or description of the same articles multiple times is unnecessary, considering our team only needs one copy of each.

After conducting the data quality assessment of the GMU Library Portal, our team concluded that this data source was not a good fit for this project. The portal contains a plethora of data that would be extremely useful; the only problem is actually obtaining the data. We planned to create a web scraper that would automate the data collection process, but during the development of that scraper, we discovered a couple of issues with the search queries in the portal. For example, we were searching for authors by their exact name, thinking that it would only deliver documents with an author matching the name exactly. However, during further investigation, we discovered that it would deliver any document with an author with the same first name or last name, not necessarily when the full name is an exact match. In addition, the search results would also include people with two last names, which is something our team is not looking for because our list of names would include people with multiple last names.

Our team's second problem with the GMU Library Portal was that there is no way to determine an author’s or a document’s academic/institutional affiliation. This was a significant issue because it could have led to multiple people with the same name being included in query results. We needed to know the authors’ affiliations to determine if we were collecting results for the intended person and not someone else with the same name. If we were to collect data from an unintended person, it would cause incorrect data to be included in our datasets and database. This data would mislead people using our product into thinking that some people are researching topics they have no experience in.

With these two issues with the GMU Library Portal, we determined that using the portal is not a viable option for a data source. As long as these two issues exist, the GMU Library Portal will never be a good option for a data source for this type of project. In this project's scope, it was much more important to have accurate data instead of having a large volume of data.

2.7 Data Collection Ethics

Web scraping is a topic that comes with some severe data ethics problems in this day and age. A person's data is a precious commodity, many people do not want it stolen, and even more corporations do not want their data stolen. Web scraping is a process that is used to collect such data. So, the question we must address is: is it ethically sound to be taking the selected data from our data sources? Regarding the first source, the website is owned by George Mason University, and our team members are students enrolled at George Mason University. The University has given our team this project, so inherently, the University is fine with our team collecting data off of an open-access website that the University owns.

As for LinkedIn, it is a social media platform that Microsoft owns and prides itself in protecting its users' data and has gotten into many legal battles with companies that are trying to collect large amounts of data from the platform. If our team collected data on people not associated with GMU, that would cross ethical lines. Since our team was only collecting data on people affiliated with George Mason's College of Engineering and Computing, we did not cross any ethical lines. The entire purpose of using web scrapers on our selected data sources was to attempt to automate the backend of our database and to not trouble the CEC staff by asking for all of the papers they have published and their research interests. These staff members are the owners or co-owners of everything they publish, and these databases only have access to them because the authors decided to add them to the databases. This thinking applies to our other data sources as well. If our team were to collect data on people not affiliated with the University, our actions would cross ethical lines.

3 Analytics and Algorithms

The algorithms and analysis used in this project were web scraping, preprocessing and consolidating data, natural language processing (NLP), and implementation of a graph database. Most of these methods addressed are the biggest challenge that faced this project: acquiring data. Web scraping was used for most data sources to extract relevant features based on patterns in source code from webpages. Preprocessing and consolidating data consisted of converting data collected from several sources into a suitable and uniform data format. Finally, NLP was used to gather keywords from large chunks of text, like abstracts and titles of journal articles, increasing the quantity of data. This section covers the four above-mentioned methods in detail.

Based on the differences in formatting between websites whose data were acquired using web scraping (namely, the GMU College of Engineering and Computing website, LinkedIn, and Google Scholar), each website required a different approach to scraping data. Additionally, the metric used to quantify the GMU faculty and researcher's relationship to research was cosine similarity, a common metric used to measure similarity between two sequences of numbers. In the context of this project, it is used to measure the similarity between a faculty member's work and a specific keyword or domain area.

The choice to store data into a graph database was influenced by a research study at University of Pittsburgh, which sought to create a directory system for recommending subject matter experts within the university. The data structure used to make these recommendations was a graph database, which drew connections between keywords and entities. This study is covered in depth in Section 1.3.1.

3.1 Web Scraping

Web scraping is the process of collecting and parsing raw data from the internet. Web scraping is vital to this project because we must collect our data from online sources. There are two ways web scraping can be completed: either with prebuilt APIs that can access the source they are designed for and retrieve the information they are designed to obtain or with self-built scrapers that a user designs to obtain the relevant information needed instead of collecting all the information. In this project, our team decided to design the scrapers ourselves in Python.

The general idea of building a web scraper consists of a couple of steps and requires a couple of packages for it to be completed; those packages are Beautiful Soup and either Requests or Selenium. Beautiful Soup is a Python library that works as a parser to extract data from HTML code by splitting the code into a parse tree. However, this library cannot request data from web servers; this is where either Requests or Selenium are used. In our project, we use Selenium because this library gives the user the ability to open a new instance of a web browser and gives the user the ability to navigate the webpage through code. This package is required because some of our sources require a login for access. In order to log in, we needed a way to send keystrokes and click buttons without doing it manually; Selenium gives us that ability. One key component required when web scraping many web pages is to call the sleep function from the time library to pause code temporarily. This is required because nearly all webpages today have a firewall that is used to block request attacks from bots designed to flood a webpage with requests in an attempt to crash the website. Pausing our code periodically and randomly will prevent the firewall from mistakenly thinking we are trying to crash the website. Once we have

gained access to the data source, we then pass the HTML code from the webpage to the Beautiful Soup parser using Selenium.

When the parser gains access to the data, the scraper can navigate the code by using functions provided by the Beautiful Soup package like “find()” or “find_all().” These two functions allow the scraper to find a specific tag in the HTML code. For example, a developer can look for just a specific tag or a class name or id within the tag. Once the scraper has obtained the tag being searched, the scraper can extract any relevant information within the tag, like a link or text.

3.1.1 College of Engineering and Computing Website

The data from the CEC website was collected using a self-built web scraper with a two-step process. The first step was to collect all the profile links of the faculty members located on the “Meet Our Faculty” page of the website. The second step was to go to each profile to collect the relevant information. The web scraper used a new instance of Google Chrome controlled by the Selenium package.

The “Meet Our Faculty” page lists ten names and titles of faculty members per page. The scraper must collect the links of the ten profiles, change the URL to the next page, collect the subsequent ten links, and repeat this process until it reaches the last page. The design of the CEC website is rather basic, meaning the source code associated with each webpage does not have an in-depth structure, and a majority of the content is located in one tag. Because of the simplicity of the webpage, all the scraper had to do was collect the tag that contained the content and use the “find_all()” function to find all of the “a” labels within that tag. An “a” label is the class used for a link in HTML code, and the “href” identifier is the location of the usable link for the “a” class. We collected that “href” identifier from each “a” class, added the links along with the person’s name to a list, and repeated the process until it completed the final page. The list of links was then saved to a CSV file.

The second step began with loading the CSV file containing the collected links. The faculty profiles are more structured than the faculty list page. The phone number, job title, and links like email and personal website are each located in separate “div” tags. However, the biography, research interests, and degrees are all together in another “div” tag. The links to email and website were collected the same way the profile links were collected in step one. The phone number and job titles were collected by going to the corresponding “div” tags and using the “get_text()” function on the tag, and it would return the text associated with the tag. In order to obtain and separate the biography, research interests, and degrees, the scraper had to move to the “div” tag associated with the needed content and use the “get_text()” function again to get all of the text associated with the tag. Figure 8 contains an example of a faculty member’s profile. The text was one long string and needed to be split by the section labels to separate the biography, research interests, and degrees into three separate strings. This process had to be completed for each link in the CSV file. Once all the data was collected, it was exported to a CSV file. This entire process is automated in the Python file, so the process can be rerun very easily if new faculty members are added to the CEC website.

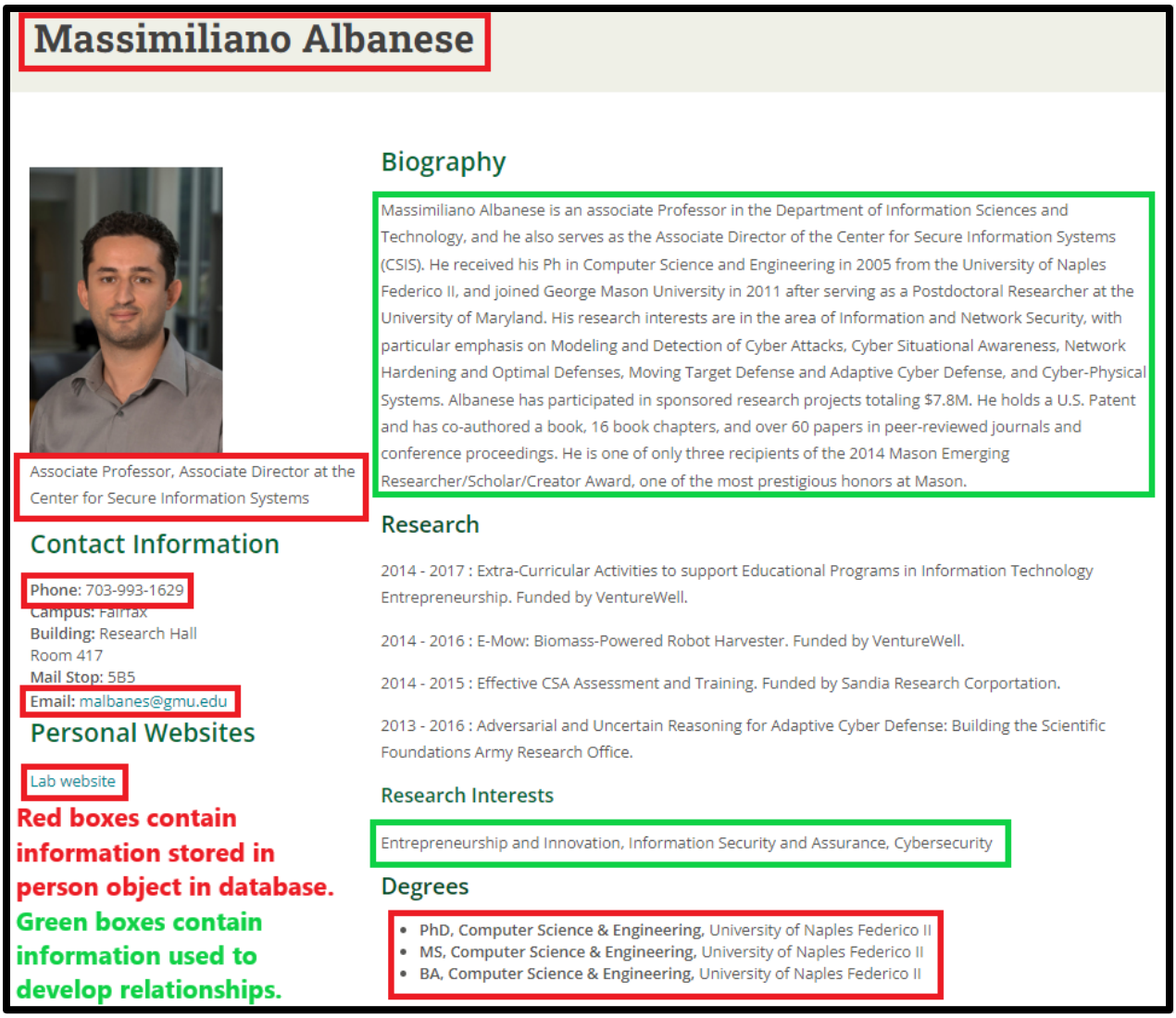


Figure 8. CEC Website Profile Example

3.1.2 LinkedIn

Like the CEC web scraper, the process of scraping data from LinkedIn begins with loading a new instance of Google Chrome. After that, the actions taken are very different. The LinkedIn web scraper is split into three parts. The first step was to collect LinkedIn profile links for every person from the list of names collected from the CEC website. The second step was to collect the "About" section from each profile that had one. The final step was to collect the title, publishing date, and description of every publication added to each profile. Before this, though, the scraper needed to log into LinkedIn. This was done by utilizing the "find_element()," "send_keys()," and "click()" functions in the Selenium library. These functions allow the scraper to find elements in the webpage, send keystrokes, and click buttons. With these functions, we could type in the username and password and click the login button.

In order to collect the LinkedIn profile URLs, the scraper had to manipulate the URL corresponding to the search page on LinkedIn. By changing the URL, the user can change the name of the person being searched. The user can even trim the results by establishing that a person's current employment needs

to be George Mason University. Establishing their current employment was important because it vastly reduced the number of possible results and greatly decreased the probability that the wrong profile would be returned. The scraper would search for each person's name in the CEC data and collect the profile link for the first result using the same process to collect links in the CEC web scraper.

In order to complete the second step, the scraper would navigate to each profile link. The "About" section for each profile was collected using the same process as the one the CEC web scraper used to collect the profile biographies. Figure 9 is an example of an "About" section you would find on a faculty member's profile. LinkedIn's webpage design implements a dynamic tag system to help prevent mass data collection. This system would change the tag names each time the page loaded, making it impossible to identify sections of the profile by their HTML tag. In order to bypass this issue, the web scraper collected all of the text from the profile using the "get_text()" function. After analyzing the text, we identified that each section would repeat the section name twice, so the "About" section would begin right after "AboutAbout" appeared in the string. This structure is the same for each section in the profile. Next, the scraper would check if the profile contained an "About" section by looking for the title phrase we identified. If it were found, it would split the text by unique section titles like "AboutAbout" or "ExperienceExperience." After completing this, the "About" section would be the second item in the split list. That string would then be passed into a DataFrame.

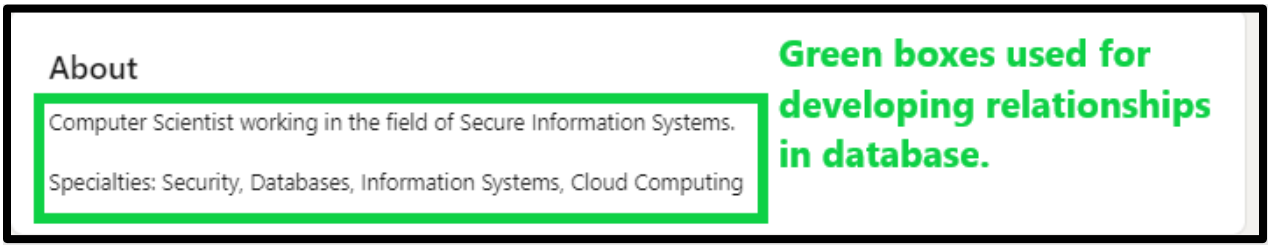


Figure 9. LinkedIn About Section Example

The third step was completed by the scraper navigating to a separate URL where the profile publications would be located if there were any. The process conducted to collect the relevant information from the publications is very similar to the process conducted to collect the CEC profile links, faculty phone numbers, and titles. The publications were in a list format, and each publication would have its own tag. The tag for each publication was collected, and for each tag, the scraper would navigate to the corresponding class for the title, publishing date, and description. Figure 10 represents the contents of each publication, but not every publication contained a description. This information was then passed into a DataFrame along with the profile owner's name. The DataFrames containing the "About" sections and publication information were then exported to a CSV file. This web scraper may not work in the future because LinkedIn constantly updates its firewall to detect bots better. LinkedIn's firewall could have caught the web scraper if it had run for a more extended period of time.

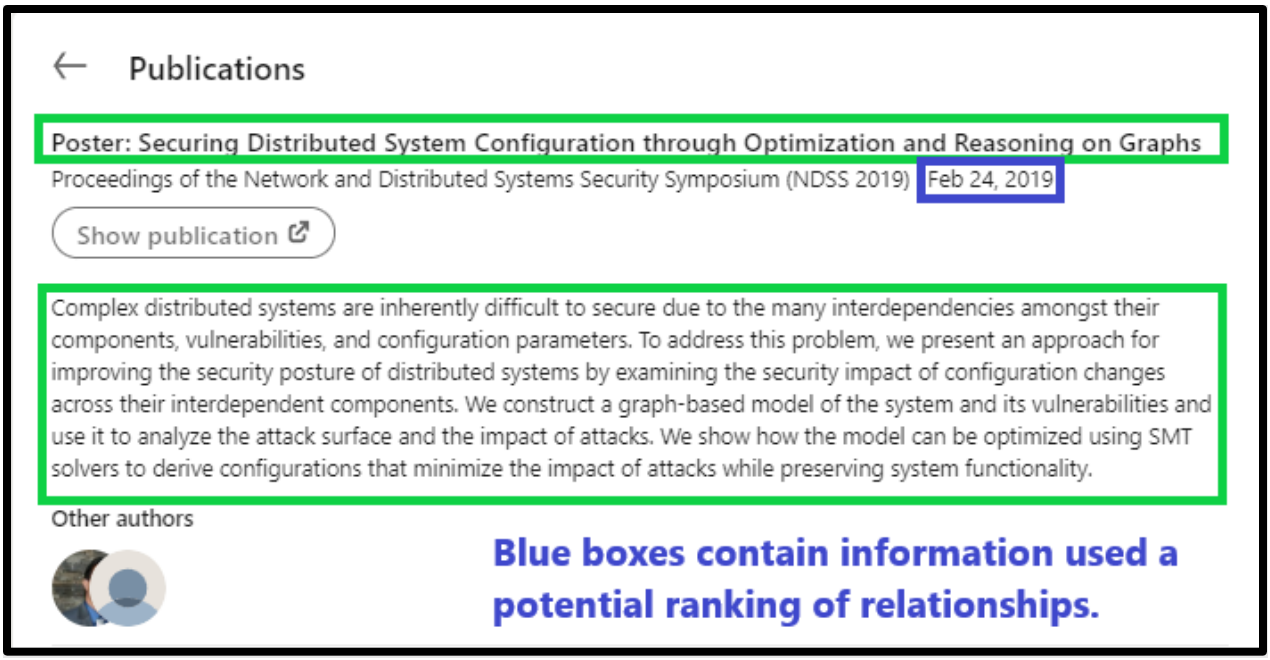


Figure 10. LinkedIn Publication Example

3.1.3 Google Scholar

Google Scholar can categorize profiles by affiliated institutions. Search results are shown in pages of ten results, sorted by the total number of citations associated with each profile. The next button is a Javascript object enabled in the page HTML code as long as there are more results to show. This feature was used to scrape data about GMU-affiliated profiles in a page-by-page manner. The data shows that about 1,677 profiles verified by a GMU email address were scraped. This list includes all the active faculty members, retired faculty, and graduate students.

The list from Google Scholar was filtered by the CEC list in different steps. First, 107 exact matches were found and stored in a new list. Then, 18 entries with the same last name were filtered according to the first letter of the first name and added to the list. In the last step, the remaining names were manually filtered, and 24 other entries were added to the list. For each of these 149 names, all the publications in each profile were visited, and titles and abstracts were scraped along with several other features. A total number of 14,696 entries were created in the resulting dataset.

Google bot detector was the biggest obstacle in this part. First, an average click time was calculated manually for each request to find a way around the bot detector. Then the sleep time in the code was drawn from a probability distribution with the same average. Also, proper request headers were designed so that the requests sent to the servers resembled those sent from a browser.

3.2 Preprocessing and Consolidating Data

3.2.1 Preprocessing

Data scraped from Google Scholar needed preprocessing. For a publication, many authors were not GMU employees; non-GMU-affiliated author names, stop words, and punctuation were removed. All letters were transformed to lowercase.

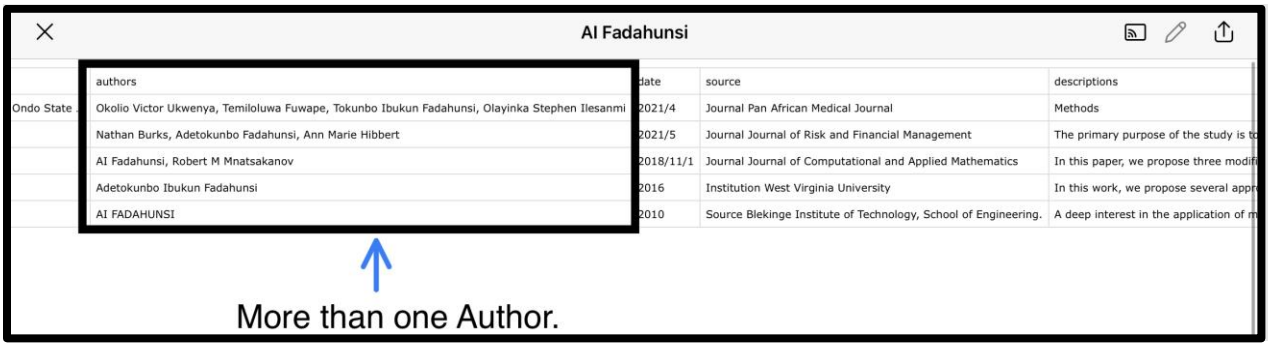


Figure 11. Multiple Authors in Publications

For Google Scholar data, authors' names from the publication's data were manually removed. After removing all the non-GMU-affiliated names, the faculty member's name was left in the author's section. Names were verified to match the ones that had been taken from the CEC website.

The NLTK toolkit was developed for Python users to deal with NLP. It offers several test datasets and different text processing frameworks. Using NLTK, a range of activities may be carried out, including tokenizing, visualization, and many more. Stop words and punctuation are typically not beneficial for information retrieval and learning; removing them reduces the number of tokens and speeds up information retrieval and learning. To remove stop words and punctuation NLTK is used. First, we had to download all the stop words using "nltk.download("stopwords")." Next, we had to indicate the language for which we wanted to remove the stop words using "stopwords.words("english")."

Additionally, we are importing punctuation from the string library at this "point.NLTK" is used to preprocess the data scrapped from Google scholar, LinkedIn, the CEC website, and Academic Search Control. Stemming is a process by which we tend to form the word stem out of the given word. For example, if the given word is "lately," then the stemming will cut "ly" and give the output as "late" this is done to find more context for information retrieval and to reduce the size of the dataset. Used stemming, changing the words' meaning as they were performed. Stemming was thus taken out of the preprocessing procedure. [23]

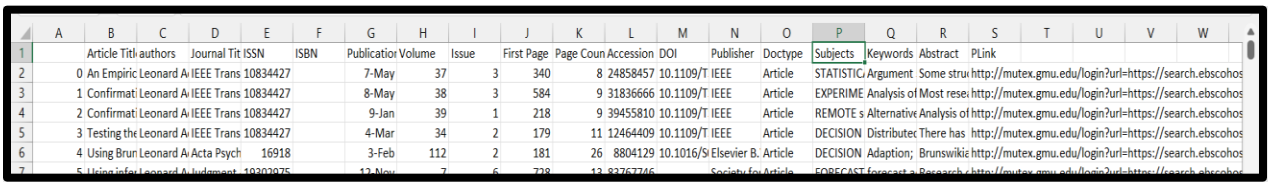
The dataset had a lot of unnecessary columns and Null values. Columns were deleted using the "drop()" function. Null values were changed to "NaN" using "fillna()," and all special characters were also eliminated.

3.2.2 Consolidating Datasets

The data extracted from Google Scholar is stored in multiple CSV files. Therefore, it was required to combine every dataset into a single DataFrame and export that DataFrame as a single CSV file.

The data extracted from Google Scholar and LinkedIn are stored in multiple CSV files. Google Scholar's data had 147 profiles of professors and their publications. Because several authors were listed in one professor's publication, the names of other authors had to be manually removed. Data scrapped from LinkedIn were divided into three different files, LinkedIn Profiles of the faculty from CEC, about sections of the following profiles, and the description, title, and date of every publication that the faculty member published on LinkedIn. The following datasets from LinkedIn are combined with each other by profile name. NaN replaced the empty cells in the data.

The "concat()" function is used to concatenate DataFrames with the columns in a different order. The resulting DataFrame would default have the same sorting as the first DataFrame. After combining data from Academic Search, CEC, Google Scholar, and LinkedIn. The DataFrames containing Google Scholar, CEC, and LinkedIn data were consolidated into a single DataFrame. Academic Search Complete's Subject, Keywords, and Abstracts are integrated into a single column and are consolidated with the DataFrame. The DataFrame is extracted to a CSV file and can be used to retrieve bag-of-words and complete other operations.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1		Article Titl	authors	Journal Tit	ISSN	ISBN	Publicatio	Volume	Issue	First Page	Page Coun	Accession	DOI	Publisher	Doctype	Subjects	Keywords	Abstract	PLink				
2	0	An Empiric	Leonard A	IEEE Trans	10834427		7-May	37	3	340	8	24858457	10.1109/T	IEEE	Article	STATISTIC	Argument	Some stru	http://mutex.gmu.edu/login?url=https://search.ebscohos				
3	1	Confirmat	Leonard A	IEEE Trans	10834427		8-May	38	3	584	9	31836666	10.1109/T	IEEE	Article	EXPERIME	Analysis of	Most rese	http://mutex.gmu.edu/login?url=https://search.ebscohos				
4	2	Confirmat	Leonard A	IEEE Trans	10834427		9-Jan	39	1	218	9	39455810	10.1109/T	IEEE	Article	REMOTE s	Alternative	Analysis of	http://mutex.gmu.edu/login?url=https://search.ebscohos				
5	3	Testing the	Leonard A	IEEE Trans	10834427		4-Mar	34	2	179	11	12464409	10.1109/T	IEEE	Article	DECISION	Distributec	There has	http://mutex.gmu.edu/login?url=https://search.ebscohos				
6	4	Using Brun	Leonard A	Acta Psych	16918		3-Feb	112	2	181	26	8804129	10.1016/S	Elsevier B.	Article	DECISION	Adaption;	Brunswikia	http://mutex.gmu.edu/login?url=https://search.ebscohos				
7	5	Using infer	Leonard A	Judgment	19302975		12-Nov	7	6	728	13	83767746		Society for	Article	FORECAST	forecast a	Research /	http://mutex.gmu.edu/login?url=https://search.ebscohos				

Figure 12. Consolidated Dataset

3.3 Natural Language Processing (NLP)

3.3.1 Bag of Words

Details of the research fields that the academics are engaged in are included in the dataset that has been compiled by our team. It contains information such as abstracts and titles that will be utilized to construct bag of words. The bag-of-words model is a simplifying representation used in natural language processing and information retrieval. Text such as a sentence or a document is represented as a bag of words, disregarding grammar. We generate the bag of words from the abstracts and titles to collect the keywords associated with each professor. Then we map these words to the respective researcher. Later these keywords will be used to search for the relevant researcher. To generate the bag of words, firstly, we must load the dataset as a DataFrame with the help of the panda's package. Since we only needed the abstracts and titles to generate the bag of words, we chose those attributes and tokenized the text into tokens. Tokenization is the process of breaking sentences into individual words represented as tokens. This package has inbuilt functions to lemmatize the tokens. We loaded the "en_core_web_sm" library from "spacy" package and collected lemmatized form of the word using ".lemma_" function. The method of lemmatization is used to change words into their root form. For instance, the roots of the words "use", "using", and "used" are all the same but are employed in various contexts. Lemmatization helps in transforming and grouping different words into their root word. The spacy also has a package "stop words" which contain a bag of commonly used words such as "the," "in," and "with." The researcher cannot be identified by using these terms. We eliminated the superfluous words from our collection of terms using this built-in bag of stop words by loading the

"default.stopwords" library from the spacy package. Once this procedure is complete, a list of terms connected with each professor is produced, and this list is utilized to map the appropriate researchers based on the keywords. The dataset is named articles dataset.

Figure 13. Bag of Words.

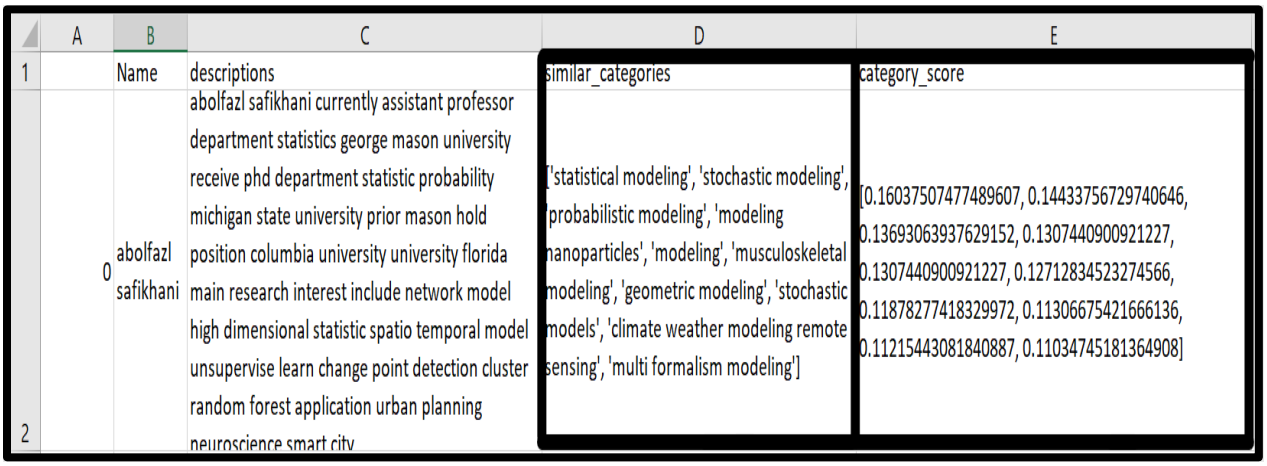
3.3.2 Associating researchers with research fields

A bag of words made from titles and abstracts of the publications cannot directly be used to find specific research fields. To address this issue, a bag of words has been made for each research topic. Python's "PyDictionary" library was used to create a bag of words for each research topic. PyDictionary is a module that can return the definition, translation, synonym, or antonym of a given word. It has three dependencies: Requests, BeautifulSoup, and goslate. It uses Google Translate for translations, "synonym.com" for synonyms and antonyms, and EordNet for definitions. First, each category, area, or topic was read into Python to create the desired bags of words. After turning all characters into lower-case, all the stop-words were removed, and each "/" or "-"character was replaced by a space. Then each remaining word was passed through PyDictionary to get a definition. Output definitions of PyDictionary are in the form of dictionaries with keys as word types and values as definitions. All the values for each input word were merged, then stop-words and punctuation were removed. All research fields and their corresponding bags of words were stored in a dataset with 583 records. This produces a dataset that contains the research field categories and their definitions.

3.3.3 Vectorization and Cosine Similarity

We made use of the generated articles, area and category datasets. The author's name and the bag of words in each article they wrote are listed in the article's dataset. There are 16109 rows in this dataset. The category name and associated keywords are included in the categories dataset. It has 556 unique rows. The area name and associated keywords are included in the area dataset. We also combined all the articles written by each author and made a dataset called combined_artciles dataset.The first step in determining the cosine similarity is to convert these datasets into vectors after loading them as DataFrames in Python. We used "CountVectorizer" from "sklearn.feature extraction.text" for this transformation. The count vectorizer creates a matrix of token counts from the collection of text documents. The count vectorizer library converts the list of words from the article DataFrame, combined_articles dataframe, area dataframe and the definitions from the categories DataFrame into vectors. After completing the vectorization process, we have four vectors: an author's vector , combined authors vector, area vector and category vector. We then calculated the angle between each author vector and all category vectors, which resulted in an array of the similarity metric of each author vector with all category vectors. Since there are 556 categories, the length of the array is 556. Then we sorted this array in descending order and chose the top 10 among them to identify the top 10 related categories to which the article belongs and selected those five. The cosine similarity between two non-zero vectors is a metric of similarity. The angle formed by these vectors is used to calculate it. To

calculate the cosine similarity, we imported the “cosine_similarity” library from the “sklearn.metrics.pairwise” package. The outcome of this process is that for every article, we could identify the top 10 categories to which the article belongs along with the scores, as shown in Figure 14. Similarly, we calculated the cosine similarities between the author's vector and areas vector and got the top 10 areas for each article. We also calculated the cosine similarities between combined authors vectors, area vectors and category vectors and got the top 10 areas and categories associated with each author. These top 10 areas and categories return the person’s possible research interests. In order to get the top 10 possible experts for each category and area, the score matrix had to be transposed so each row would represent a category or area and each column represented a person. The action was then repeated, and the top 10 people for each category and area were collected. These top 10 scores represented the ten people that had the highest likelihood of being an expert in that category or area.



	A	B	C	D	E
1		Name	descriptions	similar_categories	category_score
0		abolfazl safikhani	abolfazl safikhani currently assistant professor department statistics george mason university receive phd department statistic probability michigan state university prior mason hold position columbia university university florida main research interest include network model high dimensional statistic spatio temporal model unsupervise learn change point detection cluster random forest application urban planning neuroscience smart city	['statistical modeling', 'stochastic modeling', 'probabilistic modeling', 'modeling nanoparticles', 'modeling', 'musculoskeletal modeling', 'geometric modeling', 'stochastic models', 'climate weather modeling remote sensing', 'multi formalism modeling']	[0.16037507477489607, 0.14433756729740646, 0.13693063937629152, 0.1307440900921227, 0.1307440900921227, 0.12712834523274566, 0.11878277418329972, 0.11306675421666136, 0.11215443081840887, 0.11034745181364908]
2					

Figure 14. Snapshot of similar categories of each article along with the scores

3.4 Graph Database Implementation

Graph Databases are something nobody on our team had any experience with before the start of this project. During our research of projects and articles related to this concept of a knowledge database and directory, we discovered a project of a couple of Ph.D. students struggling to find academic advisors for their research. They wanted to find faculty familiar with the areas of research they were interested in. In order to overcome that challenge, they developed knowledge graphs using graph databases. Our team also decided that using graph databases was our best option for this project. Graph databases provide a unique way of storing and visualizing data, which is why we decided to use them. We are specifically using Neo4j to develop our database because it is one of the world’s most prominent graph database management systems, provides a free “community edition,” and grants large enough storage to store all the data required for this project. Neo4j is implemented in java, but the Cypher query language is used by Neo4j users in database development. Due to our team’s inexperience with Neo4j and the Cypher query language, our team felt that the best way to implement and navigate our database effectively would be to use Python to transform our data into an easily digestible format for Neo4j. This will reduce the number of possible complications in development.

The first step for implementing the database was passing the dictionary into the database. The structure of the original dictionary had to be transformed prior to being loaded into the database since some of the areas and categories were in list format because of their similarity. Because these listed items were not exactly the same, the team felt it was better to separate each of them into their own area or category. This process was completed using the "explode" function in the pandas library in Python. The "explode" function transforms each element of a list into its own row, where the rest of the fields in the DataFrame are duplicated for each separate item in the list. Figure 15 is an example of the taxonomy prior to the "explode" function being used on it. Figure 16 is an example of the taxonomy after the "explode" function was implemented. The new dictionary was now in an easily digestible format for Neo4j. It could be imported into the database with relationships being set between topics, areas, and categories similar to the structure of the original taxonomy document. However, by adding new categories to the dictionary, the original structure of the taxonomy broke down and left many new categories without a specific area. Our team felt this was not a significant issue due to the type of people using the database. The people using the database are familiar with the general idea of many categories and can infer the types of areas to that said categories would apply.

Digital Systems and Data	Computer Engineering	Computer Architecture
		Computing Components, Devices, and Subsystems
		Real-Time/Embedded Systems
	Computer Networks and Communications	Distributed Systems, Cloud Computing, Internet of Things
		Mobile, Wireless, 5G/Next G Communications and Networks
		Network Architecture
	Cybersecurity	Biometrics
		Cryptography
		Digital Forensics and Analysis
		Leadership, Governance and Policy
		Operations
		Privacy
		System Security Including Communications, Cyber-Physical, Hardware, and Network Systems
	Data Analytics	Data Fusion
		Data Mining and Big Data Analysis
		Visualization

Figure 15. Taxonomy Before "Explode"

B	C	D
Topic	Area	Category
digital systems and data	artificial intelligence	cognitive systems
digital systems and data	artificial intelligence	intelligent agents
digital systems and data	artificial intelligence	knowledge representation and reasoning
digital systems and data	artificial intelligence	machine learning
digital systems and data	artificial intelligence	natural language processing
digital systems and data	autonomous systems	robotics
digital systems and data	autonomous systems	unmanned and autonomous vehicles
digital systems and data	computer engineering	computer architecture
digital systems and data	computer engineering	computing components
digital systems and data	computer engineering	devices
digital systems and data	computer engineering	subsystems
digital systems and data	computer engineering	real-time
digital systems and data	computer engineering	embedded systems
digital systems and data	computer networks and communications	distributed systems
digital systems and data	computer networks and communications	cloud computing
digital systems and data	computer networks and communications	internet of things
digital systems and data	computer networks and communications	mobile
digital systems and data	computer networks and communications	wireless
digital systems and data	computer networks and communications	5g
digital systems and data	computer networks and communications	next g communications and networks
digital systems and data	computer networks and communications	network architecture
digital systems and data	cybersecurity	biometrics
digital systems and data	cybersecurity	cryptography
digital systems and data	cybersecurity	digital forensics and analysis
digital systems and data	cybersecurity	leadership
digital systems and data	cybersecurity	governance and policy
digital systems and data	cybersecurity	operations
digital systems and data	cybersecurity	privacy
digital systems and data	cybersecurity	system security including communications
digital systems and data	cybersecurity	cyber-physical
digital systems and data	cybersecurity	hardware
digital systems and data	cybersecurity	network systems
digital systems and data	data analytics	data fusion
digital systems and data	data analytics	data mining and big data analysis
digital systems and data	data analytics	visualization

Figure 16. Taxonomy After "Explode"

Our team discovered when researching graph databases that the easiest way to implement them is by passing in completely normalized data without any null values. Some of our existing data have missing values, and some other variables are in the form of lists. In order to normalize the data, we first needed to normalize the null values by either replacing them with another value or we could create separate DataFrames so each column would not have any null values. For example, we could have created a DataFrame for just the faculty members with phone numbers available on the CEC website and another DataFrame with the faculty members that did not have a phone number listed. We could then pass both of those DataFrames into the graph database separately, and the result would be that all the faculty members are still included in the database. However, some person objects would contain a phone number, and others would not. We felt that this was not the best option because if a user viewed a person in the database and the phone number field was missing, that could be misconstrued as a mistake in the database. However, if all the null values were replaced with “N/A”, it would be clear that this person does not have a phone number in the database and that it is not a mistake. Figure 17 is an example of a person object in the database.

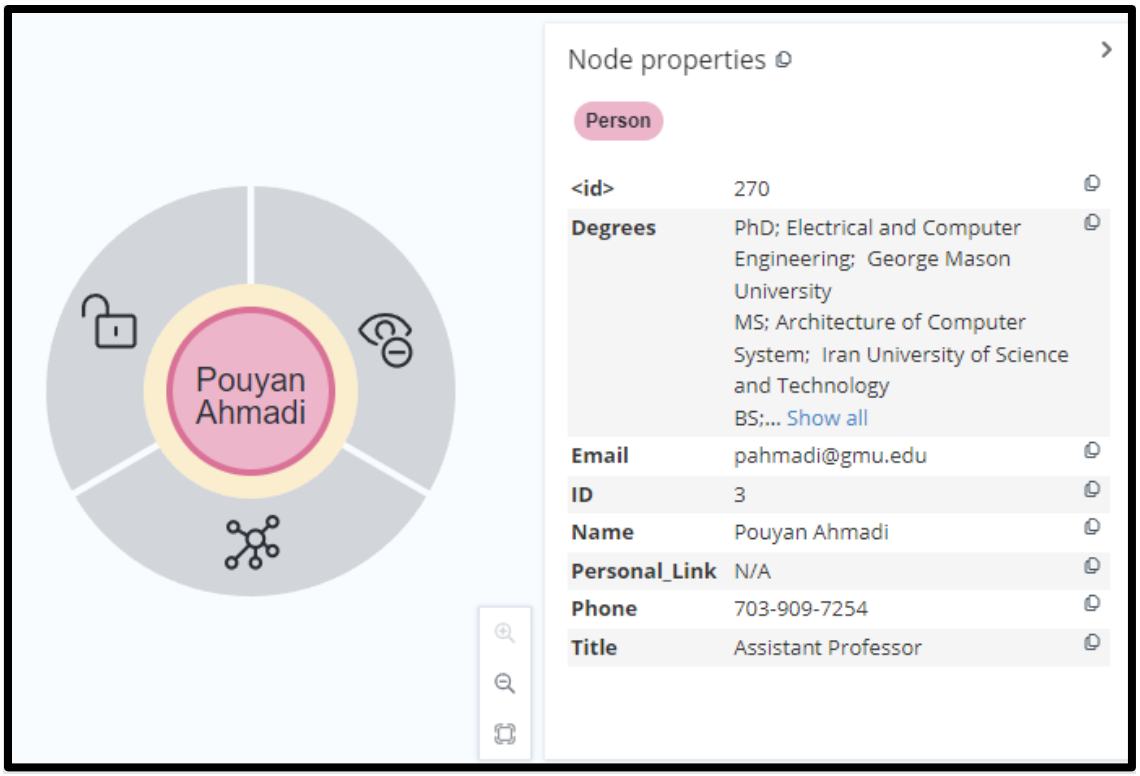


Figure 17. Database Person Object

The next step was to remove lists from our data and convert the data so that each item in a list had its own row. This aspect is vital to the research interests, considering the relationships between faculty members and their research interests are the most vital component of the database. Each interest in the list of interests would be its own relationship. In order to make these interests easily digestible by Neo4j, the data would have to be expanded in Python first. This process was also completed using the "explode" function in the pandas library in Python. Once this was completed, we could compare those interests with the research categories and areas in the dictionary and split the current DataFrame into three separate DataFrames; one DataFrame where it would only contain rows where the research interest matched a research area in the dictionary, another where the research interest matched a research category, and a final one where the research interest did not match an area or category in the database. This structure was necessary for implementation because the graph database cannot have any null values in the columns being added to the database. For the DataFrame that consists of research interests not included in the existing dictionary, those fields would be added as research categories. However, they would not have any relationship between existing areas and topics in the dictionary. Those new categories would be connected to the person we received the category from with the relationship "research_interest" and they would have a score of one considering the were collected from a profile and the relationship wasn't created using cosine similarity.

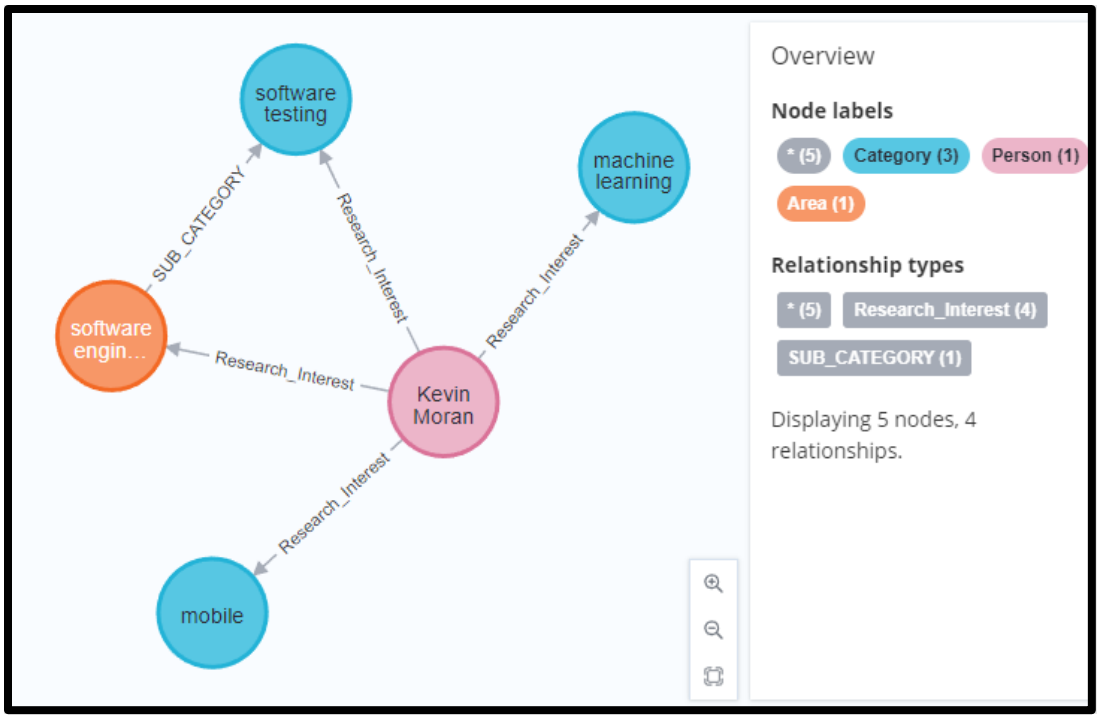


Figure 18. Database Relationships

Documents were included in the database as well. The documents come from five possible locations: CEC biographies, Google Scholar publications, Academic Search Complete documents, LinkedIn publications, and LinkedIn About sections. The documents collected from these sources were added to the database as Document objects and are differentiated by the type property added to each object. Each different source has different properties, but the properties that are consistent amongst all documents are "Doc_ID," "title," and "description." Some documents would contain the number of citations or the publishing date if they were included in the source material. The documents that have the type of Google Scholar, Academic Search Complete, or LinkedIn Publication are connected to the person they pertain to by an "author_of" relationship. A "cec_biography" relationship connects the CEC Biography documents, and the LinkedIn About section documents are connected by an "about_section" relationship.

The final step was implementing the cosine similarity scores developed during the Natural Language Processing section of Analytics and Algorithms. The cosine similarity scores measure the similarity between a faculty member's work and a specific category or area using a combination of the definitions developed using the PyDictionary package in Python and the actual category phrase. The cosine similarity scores also measure the similarity between faculty members and a specific area or category as a whole instead of comparing just one publication. All document scores developed with a value lower than 0.2, all person area scores with a value less than 0.185, and all person category scores with a value less than 0.185 were removed from the dataset to decrease the number of unnecessary relationships added to the database. These scores are added to the database as relationships between the person and area or category of research and have the label "possible_research_interest." The scores developed for an areas or categories possible experts were added to the database as "possible_expert." The scores developed for a person's work were added to the database as relationships between the document and

category or area and have the label “possible_document_subject.” The scores will be inputted as float values so a user can rank the faculty members according to their similarity to a given category to determine the best person to contact for assistance in a given research area or category.

4 Visualization

Our team developed visualizations for the cosine similarity scores created from comparing all of the documents of one faculty member with the different categories and areas of the dictionary. Bar charts, Word Clouds, and Density graphs were developed to gain a better understanding of the scores that were added to the database. We also created dashboards in NeoDash to demonstrate different options to view and interact with the graph database.

4.1 Cosine Similarity Findings

The visualizations developed from the cosine similarity scores were made to show the natural language processing techniques' results. Only some cosine similarity scores were included in the database because the relationships with a lower score were deemed insignificant and would create many more relationships that would not be valuable to the user. However, all of the results were used for these visualizations to understand better the scores' strengths or weaknesses and which areas and categories have the most relationships with faculty members.

4.1.1 Frequency Visualizations

The visualization below, Figure 19, is a bar graph showing the top 30 most frequent categories in which faculty members were labeled to have a potential research interest. The x-axis is the category's name, and the y-axis represents the frequency of that category. From the graph, you can see that model-based systems engineering appears the most, with 70 occurrences. The second most is analytic hierarchy process optimization, with around 65 occurrences. The remaining categories are ranked in descending order. You can see that the top three categories all have around 60 to 70 occurrences. There is a rather significant drop-off, with the fourth most frequent category being internet measurements, with around 35 occurrences.

Figure 20 is very similar to Figure 19, except it shows the most frequent areas instead of categories and only displays the top 20 results. The top two most frequent areas are design results clinical studies, with around 160 occurrences, and visual computing, with around 140 occurrences. Different from Figure 19, there is not a sudden drop off in occurrences in the chart. Instead, there is a gradual decrease in the number of occurrences for each area. One interesting thing that our team noticed in this figure is that the most frequent areas occur much more than the categories. This makes sense considering the categories tend to be smaller research topics. There is more than ten times the number of categories than there are areas.

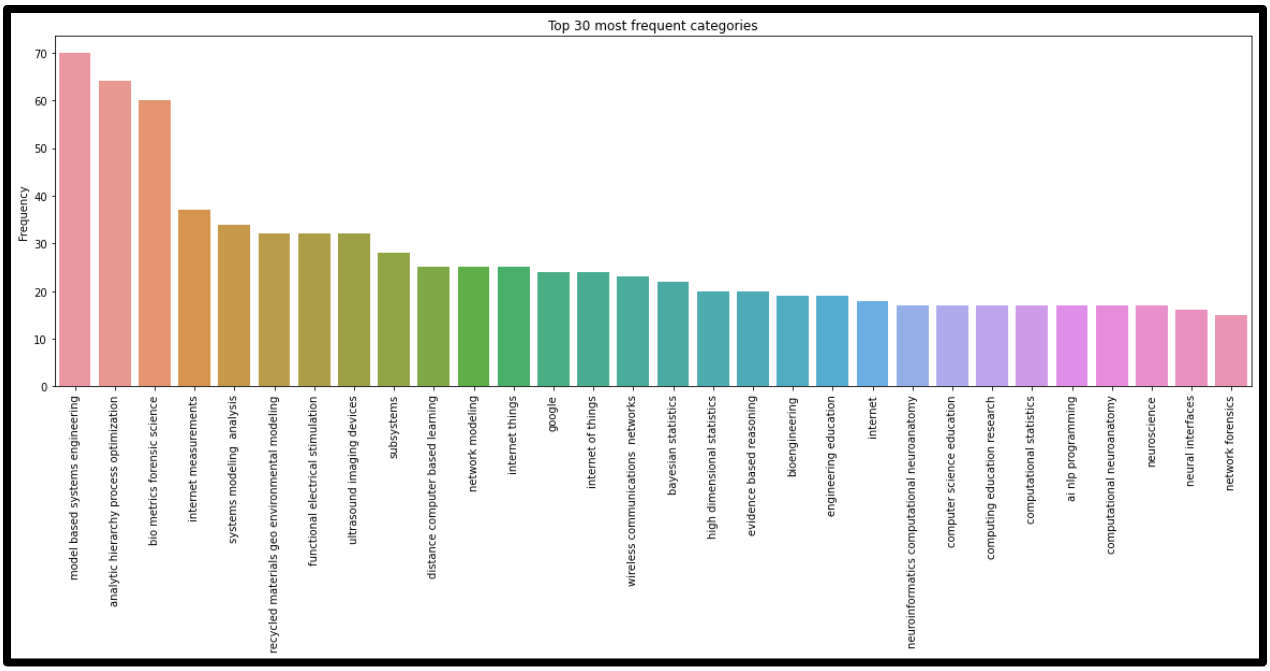


Figure 19. Person Category Label Frequency

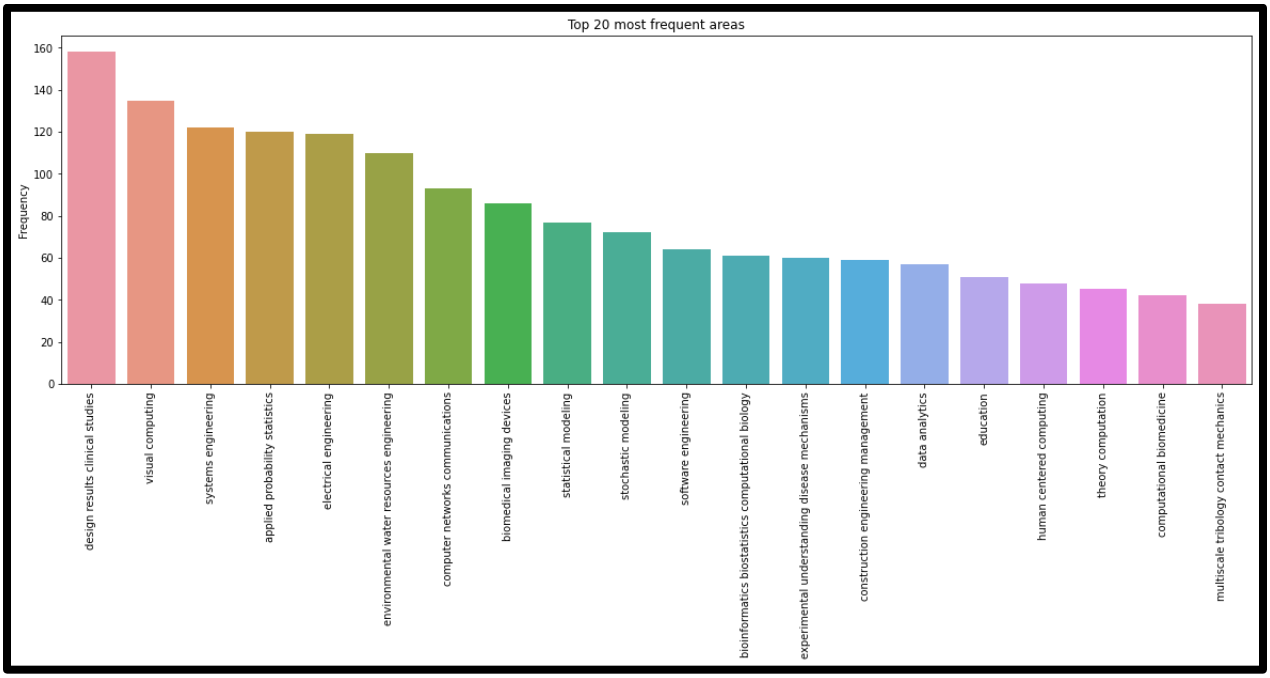


Figure 20. Person Area Label Frequency

4.1.2 Word Cloud Visualizations

A Word cloud is a cluster of different words or phrases illustrated with different sizes and colors inside a pre-defined frame. The Word cloud visualization technique is easy to explain: the more frequently a word is used in a text, the bolder and larger it is represented in the word cloud visualizations. Word clouds are also called tag clouds and text clouds. Word cloud visualization is an efficient way to extract the most prominent part of a textual data block. This type of visualization is an inexpensive alternative to preference queries which require people to fill out lengthy forms and rank their preferences in different subject areas. Much like in our construction of the graph database, word clouds remove stop words and tokenize the dataset to extract words and/or phrases.

Figure 21. Person Category Word Cloud

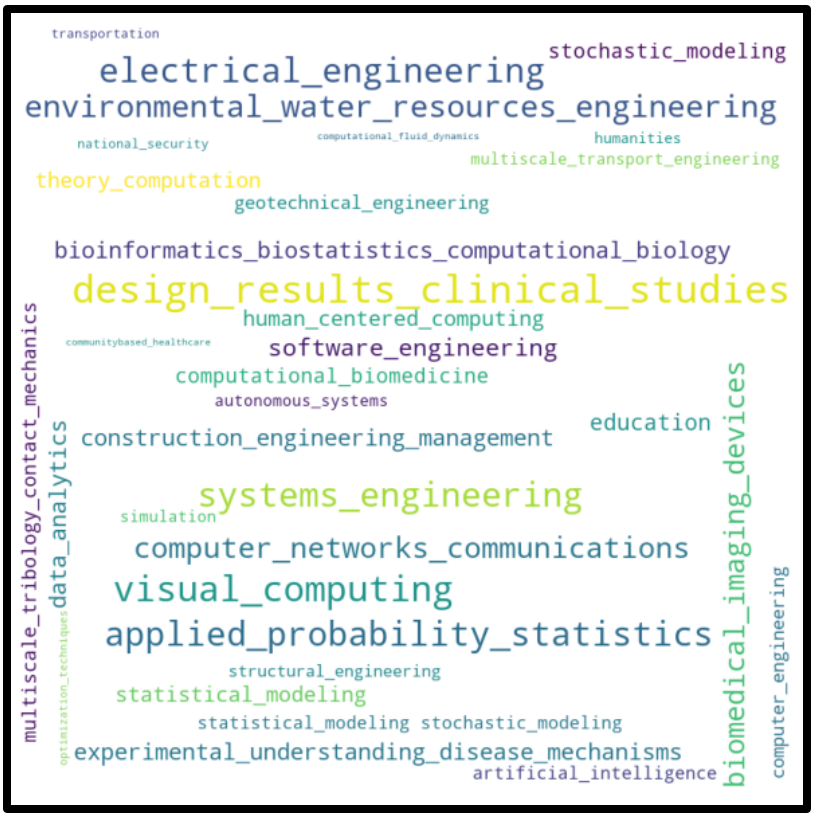


Figure 22. Person Area Word Cloud

4.1.3 Score Distribution Visualizations

Figure 23 is a density graph representing the distribution of cosine similarity scores developed from comparing the faculty members' publications as a whole and the dictionary categories. The data includes every cosine similarity score generated for the faculty members. Ten scores were generated for each member, so there are around 2,400 scores included in this graph. The graph shows that the score distribution is normal, with the average being around 0.19. However, only scores with a value above 0.185 were included in the database, so only about half of the scores used to generate this distribution were included in the database.

Figure 24 is another density graph, but this graph represents the distribution of cosine similarity scores from comparing the faculty members' publications as a whole and the areas from the dictionary instead. The data used to create the graph consists of the same number of scores as in Figure 23 (around 2,400). Ten scores were generated for each faculty member. The graph shows that the person area cosine similarity scores distribution is normal, with a mean of around 0.10. One interesting thing that our team noticed from the graph is that the range of scores generated for the areas of the dictionary is much smaller than the scores generated for the dictionary categories. The range for Figure 23 is 0.55, while the range for Figure 24 is 0.35.

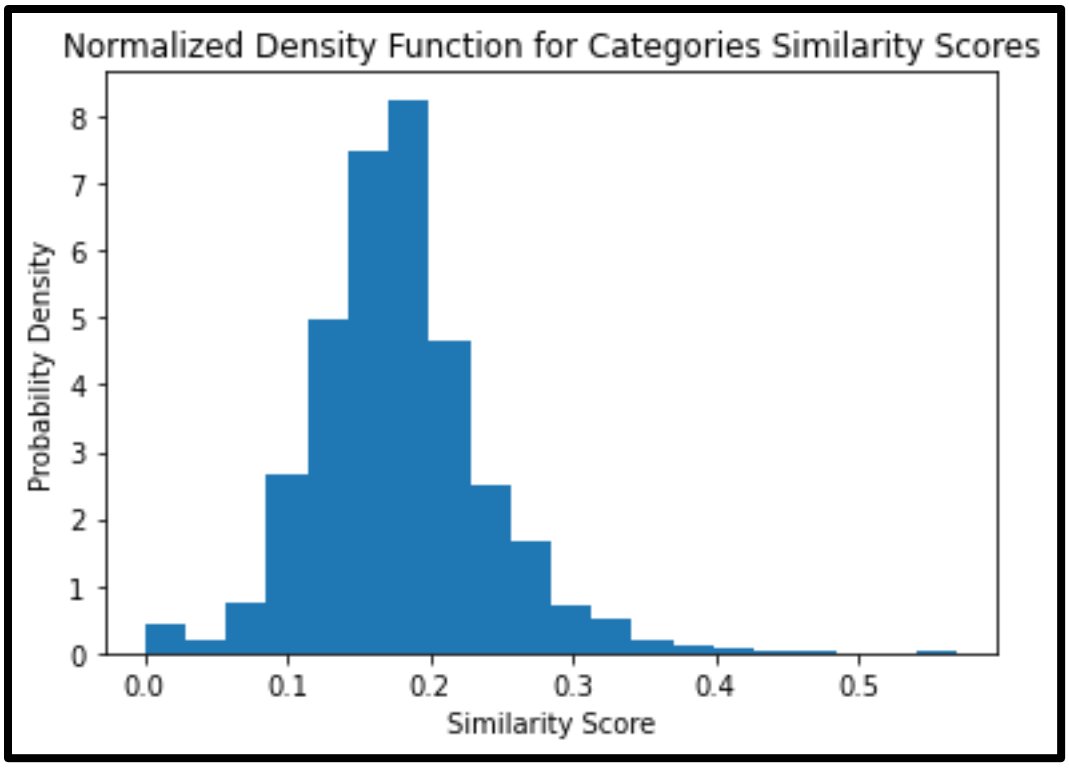


Figure 23. Person Category Score Distribution

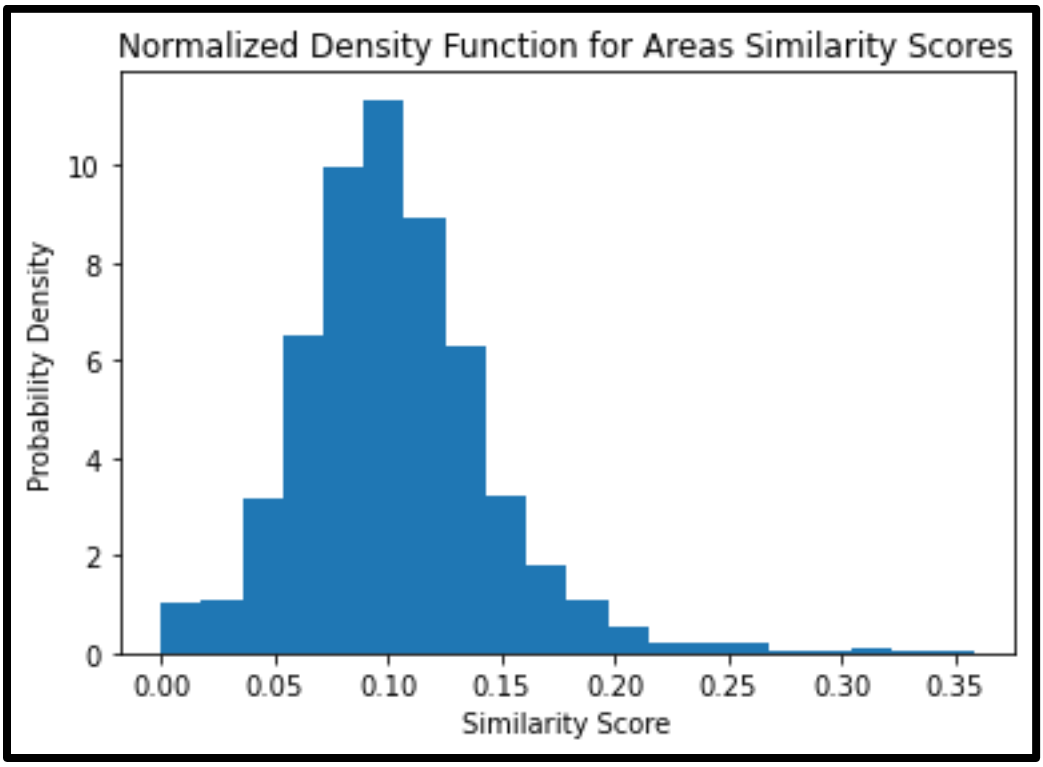


Figure 24. Person Area Score Distribution

4.2 NeoDash Dashboards

NeoDash is a good option for data visualization after creating a graph database in Neo4j since, in contrast to other front-end user interfaces, it connects directly to Neo4j. Utilizing NeoDash, it connects directly through Bolt to Neo4j and populates reports with information from Cypher query results. Interactive report options are available, and query results can be displayed as tables, graphs, bar charts, etc. It is possible to access basic styling options by altering query parameters. The connectivity may be simpler and easier to use when compared to other user interfaces like Microsoft Power BI, Tableau, Microsoft Azure, and AWS. [24] [25]

4.2.1 Database Schema Dashboard

The database schema dashboard is the first to appear when NeoDash is opened. This dashboard aims to provide an understanding of how the database is designed and what is searchable in the database. The dashboard contains a knowledge graph of the database schema. The schema labels all the possible nodes in the database and all the possible relationships the nodes have with each other. This graph is vital if the user wants to create new tables or graphs because the user needs to understand the naming structure of the relationships and nodes. After all, the cypher query language is case-sensitive, so the names of objects need to be exact to prevent errors. In addition, the dashboard contains lists of all the topics, areas, and categories of research, as well as all the people in the database. Other than topics, the other three lists contain all the items that can be searched in the other dashboards. A document list was not included because, with the sheer number of documents, a list of all of them would not be helpful to the user.

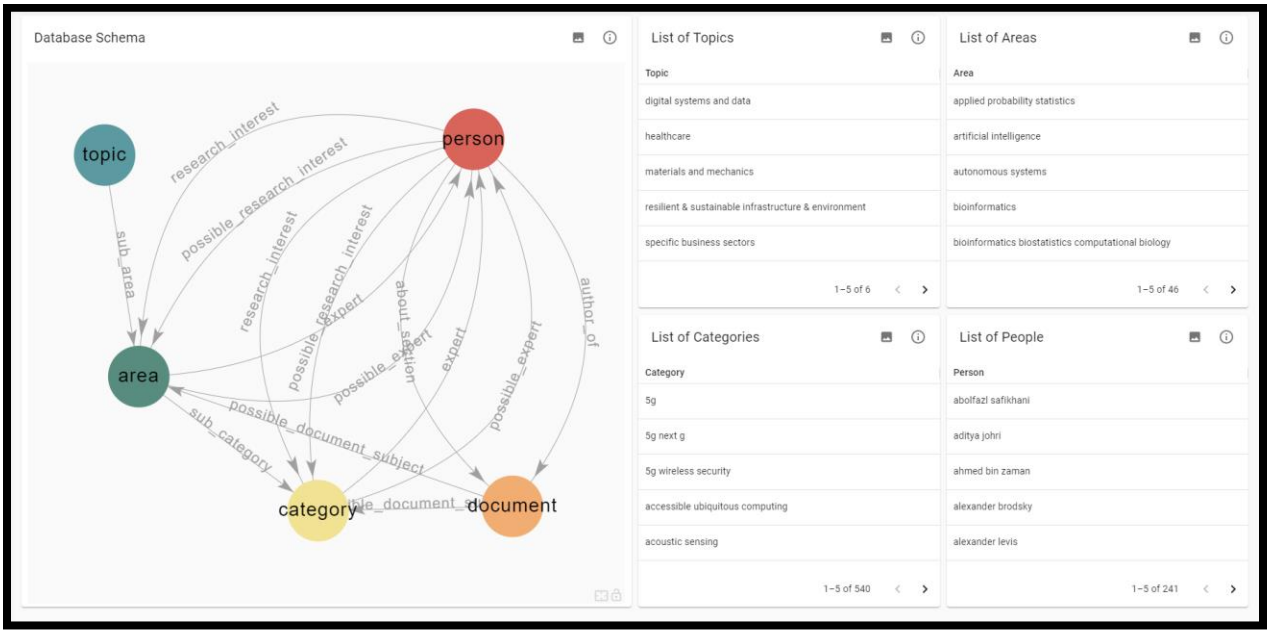


Figure 25. Database schema graph

4.2.2 Category Search Dashboard

One of the dashboards developed allows user to enter a category name and explore the people within the database with a possible research interest in the designated category or the possible experts in the

designated category. This dashboard's objective is to assist users in learning more about faculty members who are associated with that category. A dashboard with a field to select the category and display the knowledge graph related to that category is shown in Figure 26. This dashboard also contains a table which displays the cosine similarity scores of possible research interest in descending order. The table and graph for the possible research interests are then duplicated below, but instead of possible research interests, it displays possible experts.

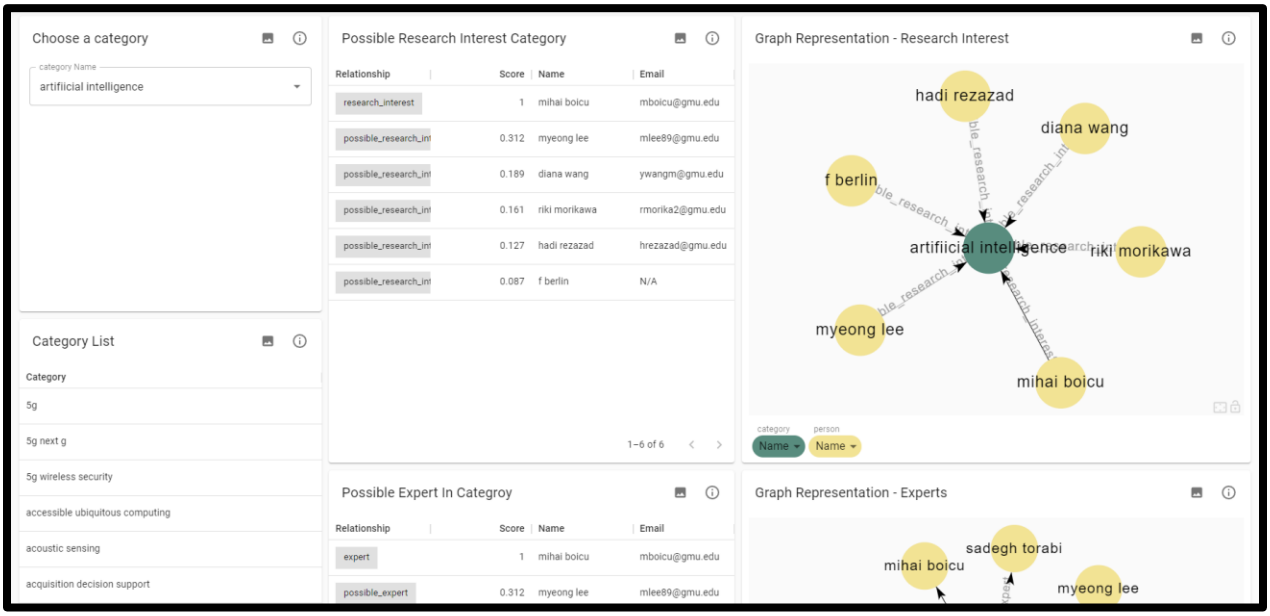


Figure 26. Category search dashboard for machine learning

4.2.3 Area Search Dashboard

The area dashboard aims to help the user find researchers in any research area existing in the database. This dashboard is based on the similarities found among each researcher's online presence. Figure 27 shows a sample area search using this dashboard. First, a user enters the appropriate area name in the "Search for An Area" block. The second block shows all the researchers interested in the entered area in a table, along with the similarity scores and each researcher's email address. The table is sorted by the similarity score; hence the researchers who appear earlier are more likely to be interested in the entered area. The third block shows a graph visualization of the resulting table. The second and third blocks are then repeated below the original blocks. However, instead of a table and graph that represent people with possible research interest in that category, the table and graph represent possible area experts. The last block shows previous collaboration between the researchers listed in the table. Mutual publications appear as mutual children nodes in the resulting graph visualization. This is specifically helpful when the user is trying to create teams of researchers interested in the same area of research by looking at previous collaborations.

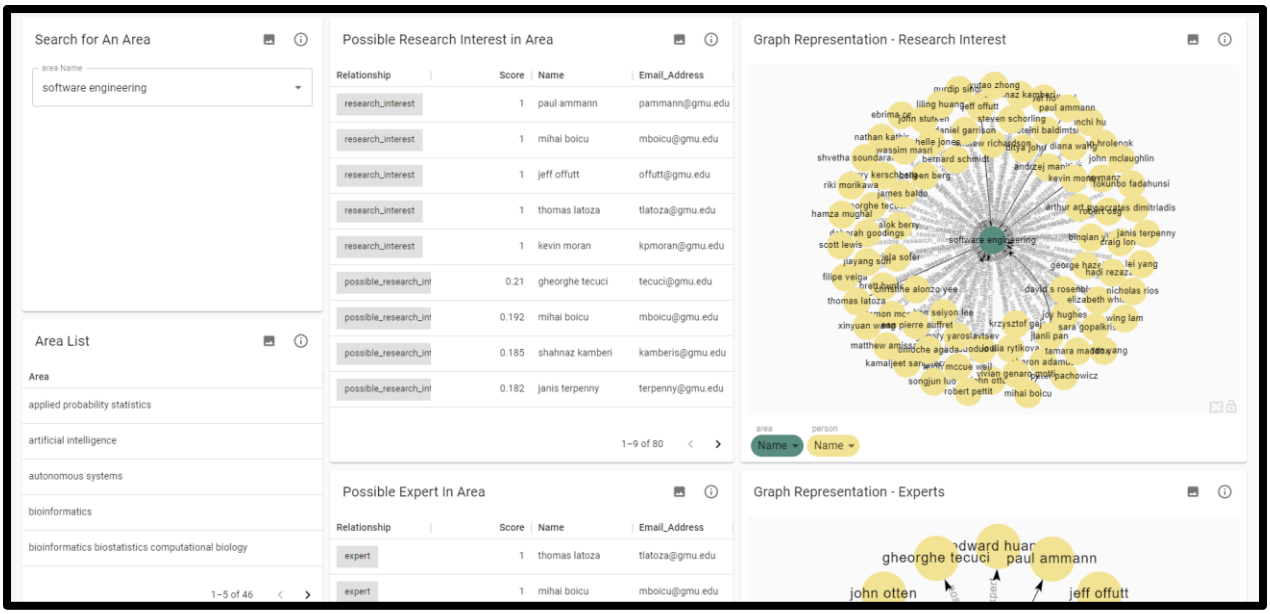


Figure 27. Area search dashboard for software engineering

4.2.4 Name Search Dashboard

One of the dashboards created was one that gave a user the ability to search a faculty member's name and view their degrees, email, titles, categories of interest, categories of expertise, documents, areas of interest, and areas of expertise. The purpose of this dashboard is to help a user learn more about a faculty member that was searched for specifically. Figure 28 shows a dashboard with a table of academic-related information, cosine similarity scores with associated areas sorted in descending order based on score, and the knowledge graphs of categories of research interest, and categories of expertise. This table and graphs are then repeated below in the same dashboard but, instead of representing categories, they represent areas. This entire dashboard can be populated by entering a specific name into the "Search for a name below" block.

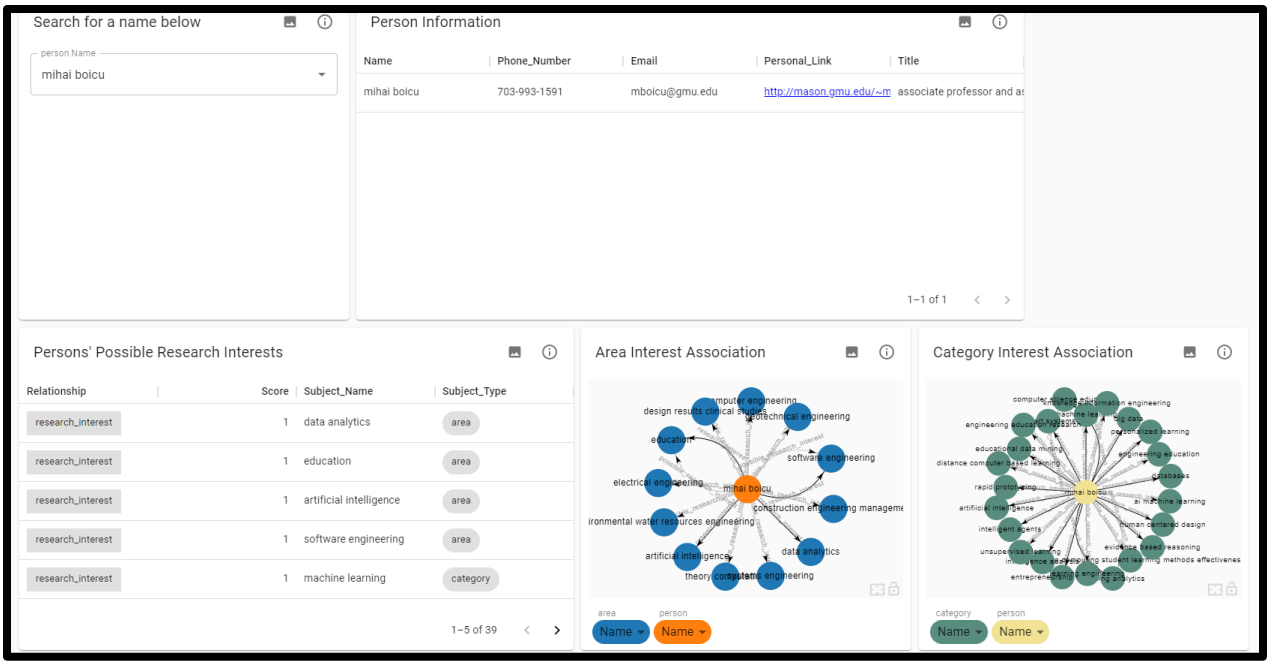


Figure 28. Name search dashboard for Mihai Boicu

4.2.5 Document Search Dashboard

The document search dashboard is designed to allow the user to search through any document in the database. The dashboard allows the user to search through all of the documents that are related to the selected category, area, or person. The way the dashboard searches for documents related to category, area, or person is conducted similarly and displayed in that order. When searching for documents related to a specific category, the dashboard contains three items that display relevant information. The first item is a single value block that identifies the category used to create the Category Table and the category Knowledge Graph. This category comes from the parameter select block in the Category Search dashboard. The second item is the Category Table, which displays all documents related to the selected category. They are ordered by the cosine similarity score of each document with the selected category. The table also contains the document's title, the document ID, and the document's author. The third item is the Category knowledge graph, which displays the information in the table in a graph format. The documents are labeled by their IDs, while the category and authors of the documents are labeled by their names. These three items are repeated for the area searched in the Area Search dashboard, and the person searched in the Name Search dashboard. While the area table and graph are the same as the category, there are a few changes to the name table and graph. The Name table contains the subject type, the subject name, the cosine similarity score, the document name, and the document ID of each document. The document's author was removed from the name table because each document has the same author. The name graph also differentiates the type of each subject in the knowledge by node color.

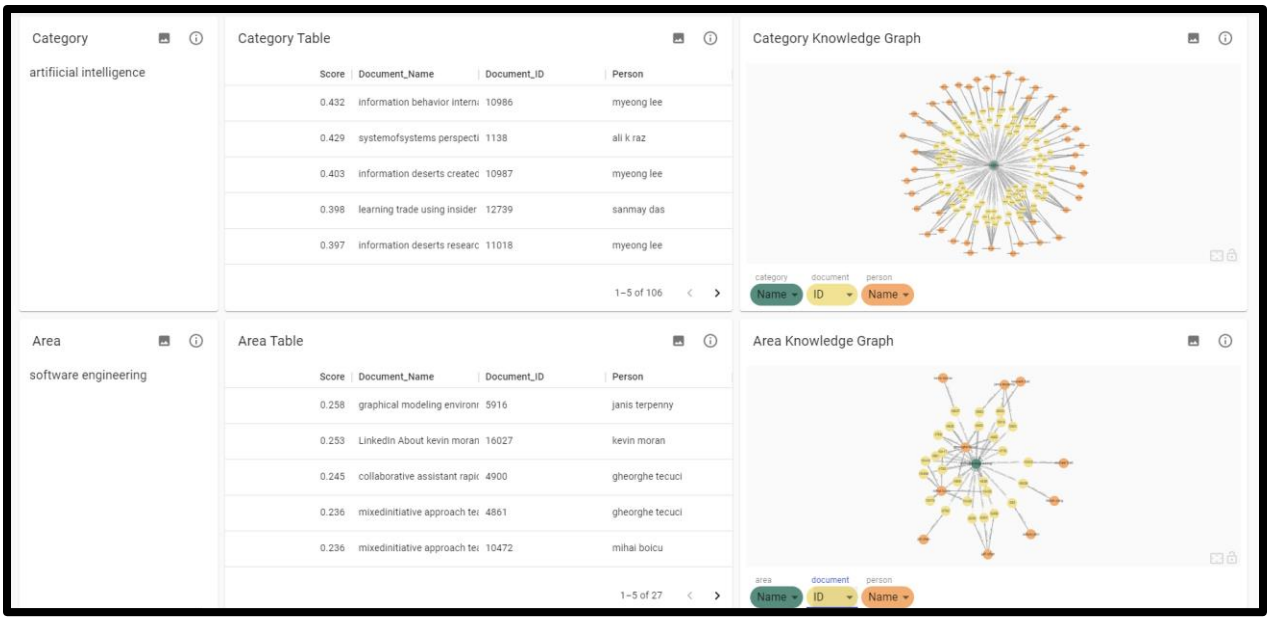


Figure 29. Document Search Dashboard Part 1

The document search dashboard's final part contains items to display information about a specific document. The first item is a parameter select figure that allows the user to search for a specific document by ID. Once a Document ID is selected, the Document Possible Subjects table, Document Knowledge Graph, and the Document value block will populate with the relevant information about the document. The table contains the subject type, subject name, cosine similarity score, author, document type, and publication date. The author, document type, and publication date will always be the same for each row because only one document can be searched at a time. However, the subject type, subject name, and cosine similarity score will change for each document's relationship. Next, the knowledge graph will populate with the document's relationships, including the author and possible document subject. Finally, the Document value block contains a document object that, when hovered over, reveals all of the information that the specific document object holds in the database, like the description or citations, if there are any.

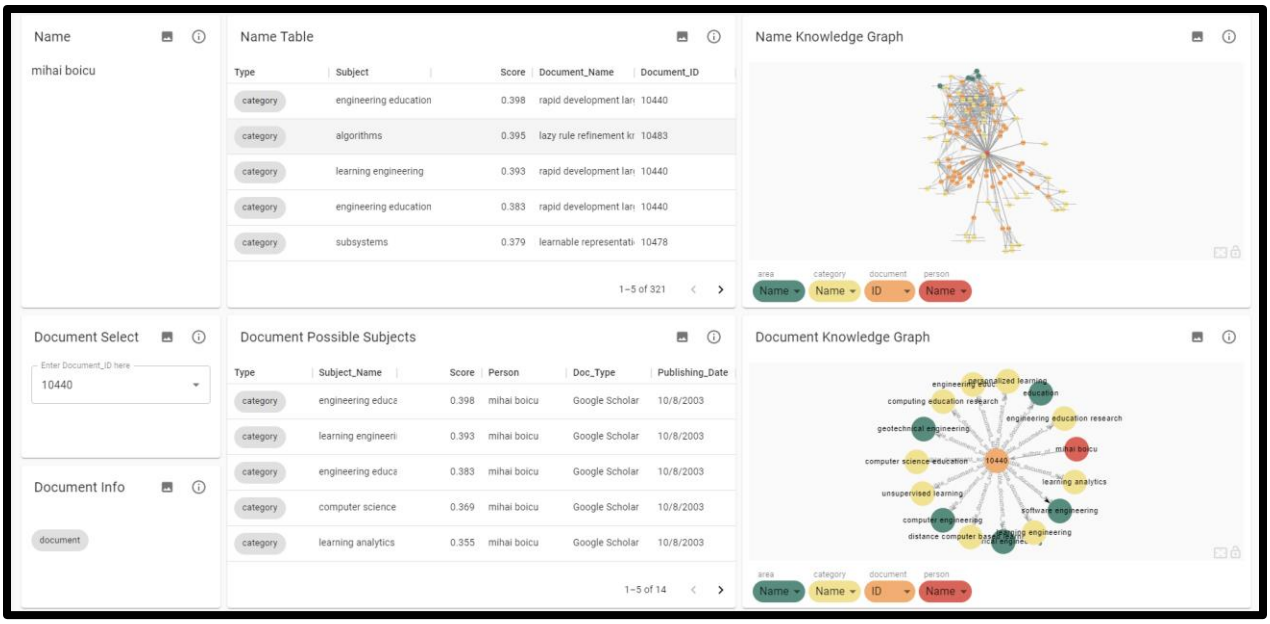


Figure 30. Document Search Dashboard Part 2

5 Findings

We developed a dashboard that lets users search for relevant subject matter experts by providing area, category, specific name, or published content. Once the user fills the search bar with the field of their choice, the dashboard will generate suggestions. The suggestions are presented in a graph format with nodes and relationships, allowing the user to find the expert they are looking for easily. Furthermore, the user can hover over the nodes to get more information about the person they are interested in. The dashboard also generates a table containing information about the author's relevant publications by associating it with the cosine similarity scores. This aids the user in determining how relevant the work is to a particular topic or category.

We encountered multiple obstacles while working on this project:

1- Finding names of GMU CEC researchers:
We spent a lot of time trying to find a reliable list of all the researchers who work for CEC; however, this task proved to be a difficult one. The most reliable solution to this issue is scraping each department and research center's webpage for the names and other information about their faculty members and researchers. This solution, however, is not feasible simply because the format of those web pages is not uniform, and the scraping operation has to be done manually to some extent. We finally compromised to the list that we found on the CEC website.

2- Name discrepancies
The research history of each researcher was extracted from websites like Google Scholar and LinkedIn. These websites index items based on the profiles built by researchers. People also do not necessarily use their full legal names when building social media profiles. For example, some people use their middle names or nicknames. This problem is even more highlighted when it comes to non-English names.

3- Robot blockers
Although indexer websites like Google Scholar use web crawlers to build and update their databases, they do not like to have their data collected using the same process. Although it is not unbeatable, the Google Scholar bot detector is robust and updated frequently to block automated data collection.

6 Summary

This project sought to provide George Mason University with a tool that would allow someone to find a research partner with expertise in a domain area. Previous research found that storing researchers' data scraped from online sources (e.g., Google Scholar) in a graph database was an effective approach to forming relationships between researchers, areas of expertise, and published content. In order to populate the graph database, this project relied on self-built web scrapers and natural language processing used on the following online sources: George Mason University's College of Engineering and Computing website, LinkedIn, and Google Scholar. This project also used data obtained from Academic Search Complete through a brute-force data collection method. The data stored in the database included researchers' names, contact information, related categories, related areas of expertise, and published content. Moreover, the database was populated with keywords identified by natural language processing and cosine similarity scores, indicating their work's relevance within an area or category. Neo4j, a software platform known for facilitating data storage in graph databases, was used to manage the graph database.

The end product was a front-end user interface called NeoDash (developed by Neo4j), allowing users to look up relevant subject matter experts by area, category, specific name, or body of published work. By combining it with the cosine similarity scores, the dashboard also creates a table that lists the details of the author's pertinent publications. This helps the user assess the work's relevance to a given topic or category. For example, suppose a user was interested in finding researchers with experience in artificial intelligence. In that case, NeoDash will provide a list of researchers who have published work in the field of artificial intelligence. In some instances, a user may be interested in searching for the top 10 experts in an area or category based on their associated cosine similarity score.

While working on this project, we ran into numerous challenges. Although we spent a lot of time looking for a trustworthy list of all the researchers working within the CEC, this effort proved challenging. Each researcher's past research activities were taken from LinkedIn and Google Scholar websites. When creating social media profiles, people do not always use their complete legal names. This issue was much more evident regarding names that were not in English. Although indexer websites like Google Scholar use web crawlers to create and maintain their databases, they do not enjoy having their data collected in the same manner. The Google Scholar bot detector is quite effective and is frequently updated to prevent automated data collecting. This was also one of the challenges.

Although the end product is not complete, this tool has great potential to become a staple in George Mason University's ability to conduct a high level of research across all colleges. The dashboard developed is a major steppingstone in providing researchers at George Mason University with a tool to look for research partners easily.

7 Future Work

The database and front-end tool developed is a valuable tool for faculty members to create research teams and find research partners. Even though the database is usable and will give users accurate information, a few things still need to be refined and added in the future. The additions will give the user a cleaner and more expansive product that is hopefully usable for every college in the University.

The first item that should be addressed is the dictionary. The dictionary is a hierarchical structure design based on the taxonomy of the CEC, which the CEC initially used to keep track of all existing research in the CEC. The structure is split into three buckets: topics, areas, and categories. All topics have sub-areas, and all areas have sub-categories. More categories were added to the dictionary using research interests collected from the CEC website and Google Scholar. The hierarchical structure deteriorated by adding these new categories because we did not create any relationships between existing areas and new categories. The new categories were added but unattached to the original structure. Some of the added categories would be better fit as areas instead of categories. Categories are supposed to be smaller subsets of areas, but some of the collected research interests are broader areas and do not fit the definition of a category. Even though every category and area were given scores for possible experts the same way, the cleanliness of the database and front-end would benefit if every added research interest was in its proper bucket. This issue does not deteriorate the results given by the product very much but creates a less clean database.

The second item that needs to be addressed is the type of people included in the database. The people included are teaching faculty in the CEC because these are the only people listed on the CEC website. Our team needed to receive information about faculty members that only did research or members of organizations connected to the CEC, like the C4I and Cyber Center. The only way to obtain that information is through the CEC administration staff; we could not receive it throughout this project. Adding research faculty members would significantly increase the number of viable research partners in the database. Of the 240 faculty members listed on the faculty website, only 125 had published articles on research they have conducted. This is because teaching faculty members' primary focus is teaching, and not all faculty members are required to do research, while research faculty members' only purpose is to conduct research. The research faculty members fit this project's scope much more than the teaching faculty members, which is why it is vital in the future of the product that they are added to the database.

The third item that needs to be addressed is the project's original idea. The original idea for this project was to make this tool for the entire university, not just the CEC. Our team, along with the insight from Dr. Wells, decided to reduce the project's scope just to include the CEC to show proof of concept. The CEC was also the most crucial college to Dr. Wells, considering the CEC is the college where he conducts most of his research. Expanding this product to all the remaining colleges in GMU is still a massive undertaking. It requires the collection of teaching and research faculty members in every college and the collection of their written publications. These publications may be on Google Scholar but are not guaranteed, and because of this, new data sources would most likely have to be used, and each new data source would require a new web scraper. The scores, database, and front-end are developed, so there would be no additional required work for those steps. The focus and problem with expansion is the collection of data, not the development of a database.

The final item that needs to be addressed is another original aspect of this project but must be removed from the scope due to the time constraint and the number of issues that occurred when trying to accomplish this goal. The objective of making the database fully self-automated was a daunting task. It was made impossible within the time restriction because our team would need to develop a solution to get by two-factor authentication. Another issue that caused our team to remove the product's automation was how our data sources displayed names. Some sources stored names as the first name, then the last name, and some other sources stored names as the last names, then the first name. Some sources did not use full names, and some used the middle initial or even the full middle name. There was no centralized structure for displaying the names. Our team had to manually go through all the names and make sure they were consistent in every data source; if they were not, they would need to be manually adjusted, so they matched the names from the CEC website. Developing a way to automate this process would significantly reduce the number of person-hours maintaining this database and could potentially be fully automated.

8 Appendix A: Code References

8.1 Link

https://github.com/bzevin/Team-Demystifiers-DAEN-690.git

8.2 Project Description

One of the frustrations here at the GMU C4I & Cyber Center has been not being able to find people rapidly who are working on related topics, be they AI, cybersecurity, European politics, community resilience, etc. Mason tried to assemble an "experts" list a few years ago based on keywords, but it has been hard to keep current.

The project consists of four elements, data collection, natural language processing, database implementation, and database visualization and reporting. The data collection is completed using self-built web scrapers for specific sources. Natural language processing is completed using Python to remove stop words and conduct cosine similarity to create scores between faculty members and research that represent how much research they are conducting in that area. Next, the database is implemented in a graph database format due to its unique ability to visualize data. The graph database will be implemented using Neo4j because it is one of the world's largest graph database management systems. It provides a "community edition" free and grants large enough storage to store all the data required for this project. The final element of this project is to visualize the database in a simple and informative way to the user. This process is completed using NeoDash, a reporting tool very similar to Tableau or Power BI. However, it is unique because it is an open-source project designed specifically for Neo4j and graph databases. This tool allows our team to create dashboards like Tableau but maintains a graph database visualization style.

8.3 Project Goals

A good data analytics problem for students might be to write a bot that would check out (a) bios for Mason faculty and researchers, (c) Google Scholar, ResearchGate, etc., to assemble lists of people working in particular areas. Some kind of Institutional Review Board (IRB) approval might be needed, but all the material would be public domain so it should not be a problem.

The optimal outcome of this project was to create a working prototype that could be re-run periodically to keep the roster of faculty and their research interests updated without burdening the scholars; due to unanticipated setbacks and problems with the standardization of the data collected. The project's goal shifted to making a working database that could accurately label the relationships between researchers and their interests that would have to be updated manually and would need to be maintained by an individual or team.

8.4 Project Deliverables

- Showcase Presentation & PowerPoint Slides
- Final Project Report
- GitHub Repository for data and code artifacts
- Working Prototype

8.5 Repository Contents

The GitHub repository contains all the files not restricted by the university or LinkedIn. The original taxonomy received from the university is not included, and the collected About sections and publications found on faculty members' LinkedIn profiles are not included. The repository is split into five folders:

- Data Cleaning: This folder contains all the files used for data cleaning or created from the data cleaning process.
- Data Collection: This folder contains all the data collection files. This folder is where the self-built web scrapers are located.
- Data: The folder contains all the data collected from the web scrapers or the final data generated through the data cleaning process and natural language processing.
- Database Files: This folder contains all of the files that are used to transform data to be passed into the database, the transformed files, and the database itself.
- Natural Language Processing: This folder contains all the files used to complete the Natural Language Processing, such as the stop word removal and the cosine similarity.

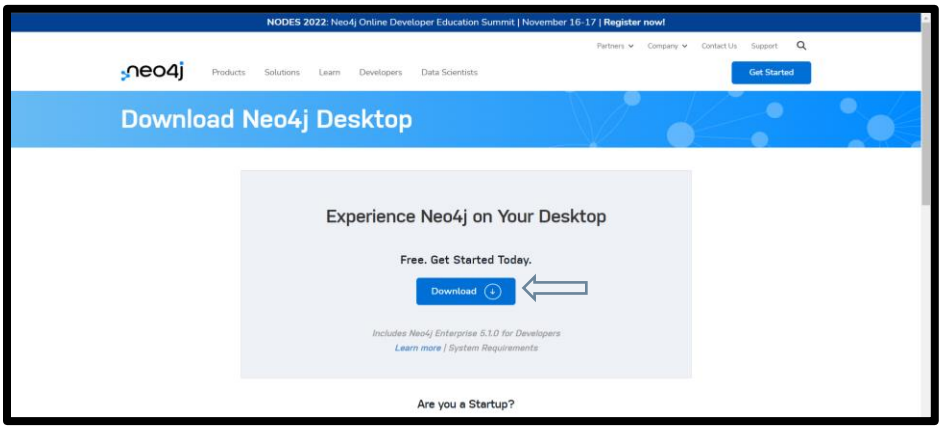
9 Appendix B: Installation Guide

Below are the steps necessary to be able to interact with the front-end interface we developed. These directions cover all steps needed, from installation to interacting with the tool.

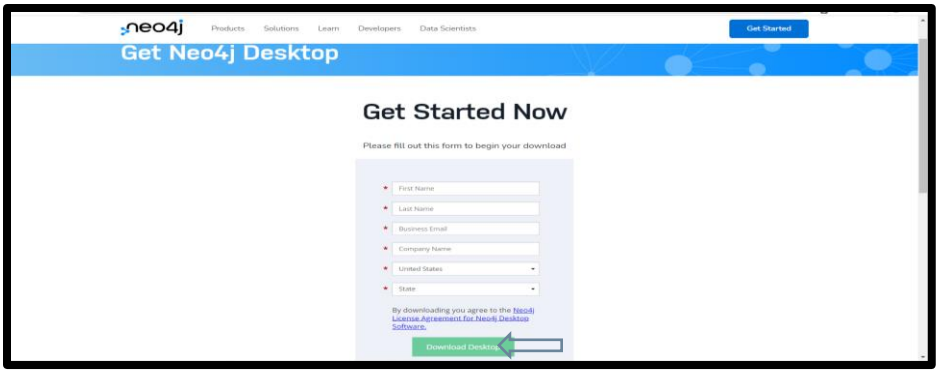
9.1 Neo4j and NeoDash

STEP 1-

Open Web browser and search for https://neo4j.com/download/ , Click on Download [26]



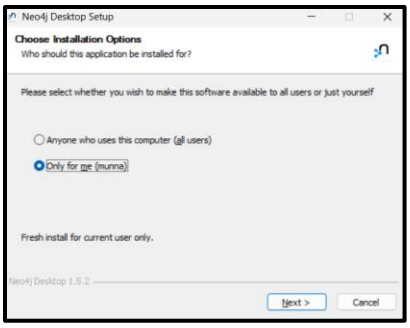
Fill in the details and continue to download.



On the next page, make sure you follow the instructions on the page. Make sure to copy the activation key.

Step 2-

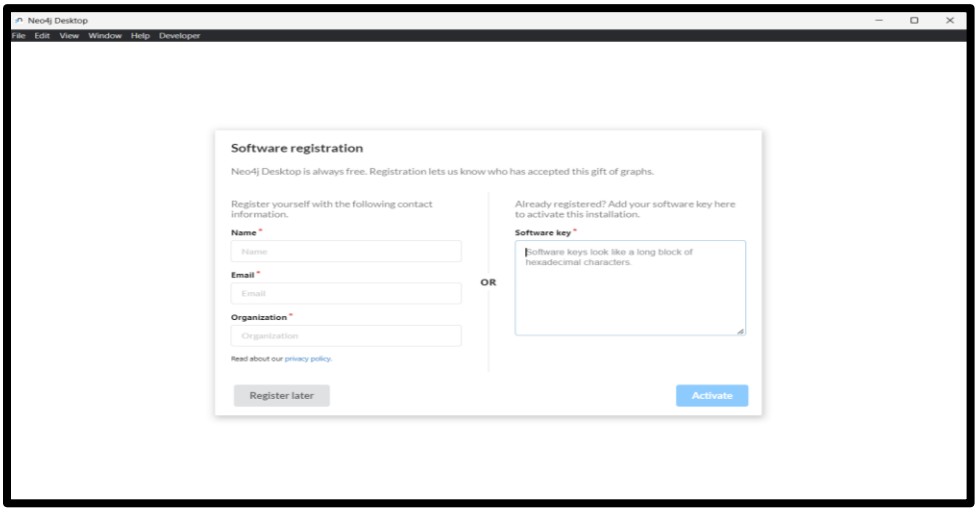
Start installing Neo4j downlaoded file.



After completing the installation, Run Neo4j.

Step 3-

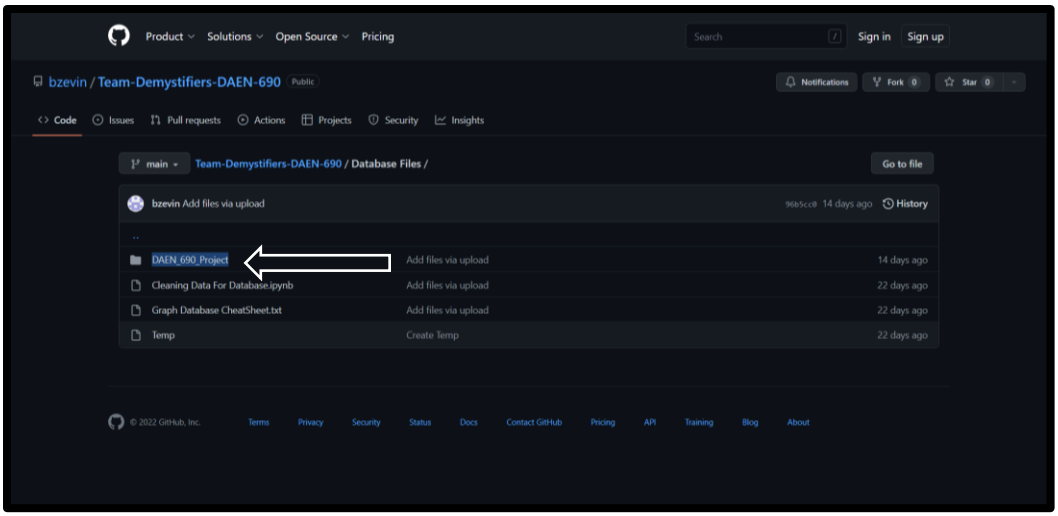
After running Neo4j, Paste the activation key which was copied from the web.



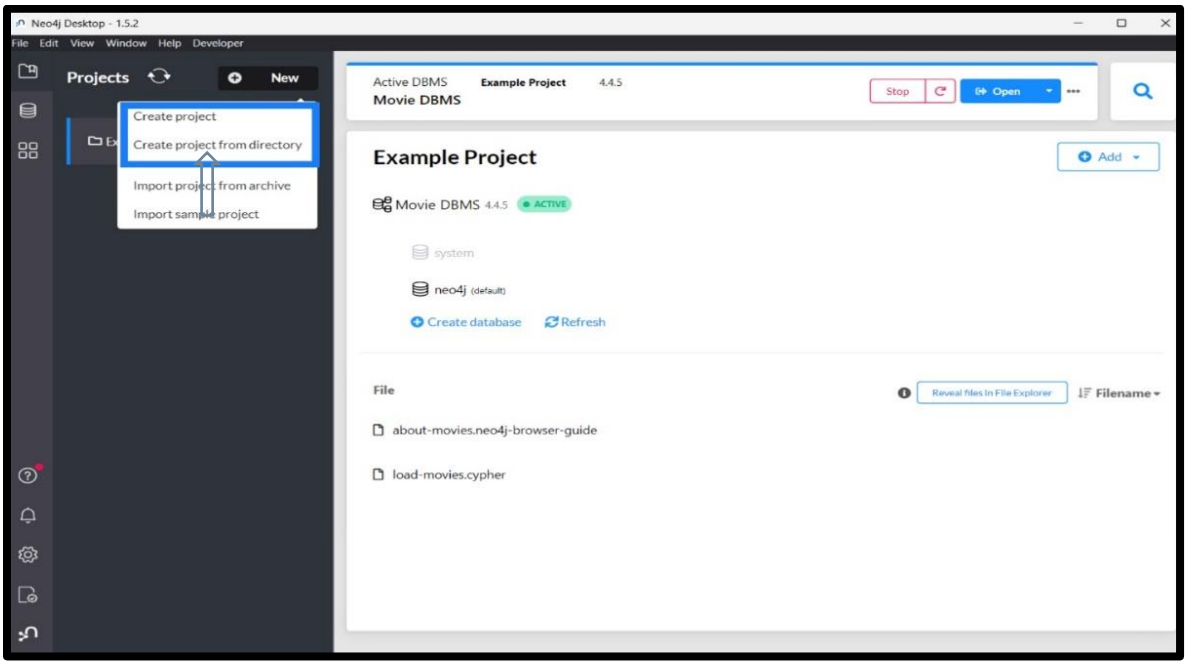
Step 4- Creating Project or uploading an existing Project

The project file can be downloaded from the GitHub repository. **To access GitHub repository**([27])

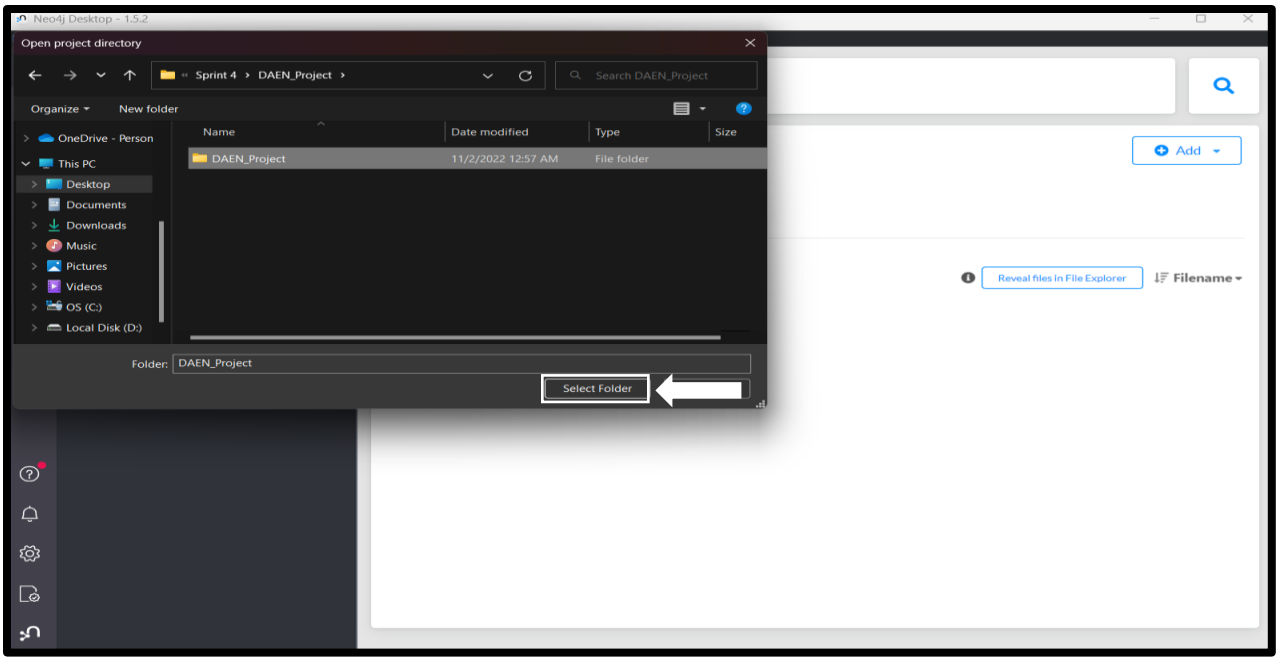
Download the GitHub repository and select the database folder. Then you can access the DAEN_690_Project folder.



After downloading the GitHub repository, to create or upload an existing project click on new and select Create project and select Create project from directory.

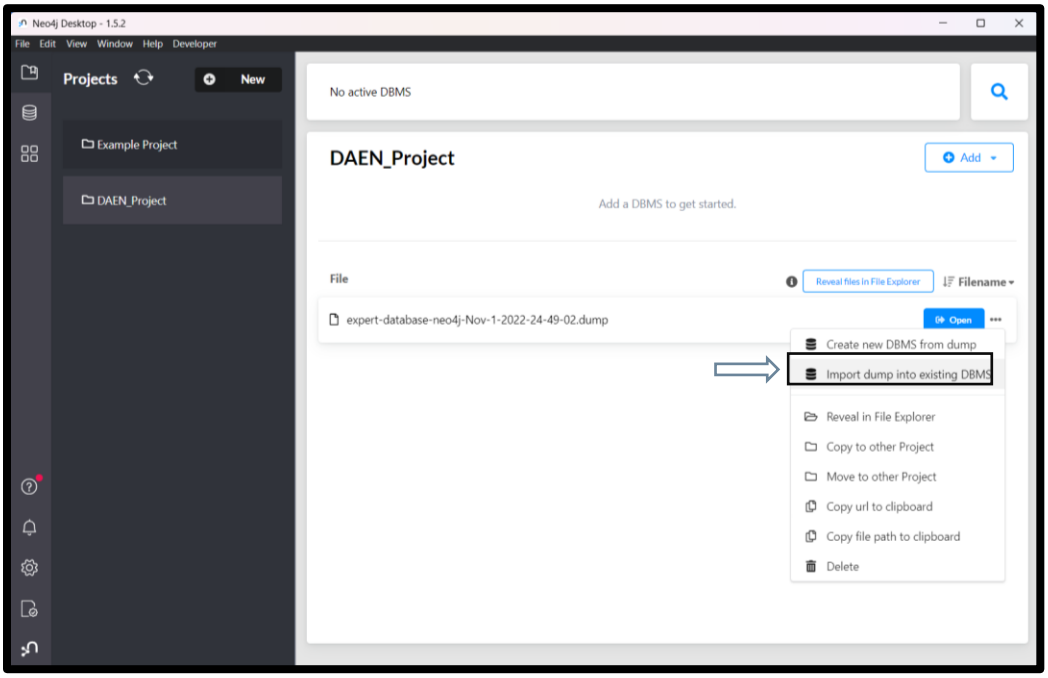


Select the project folder which is downloaded from GitHub and then select the folder into Neo4j.

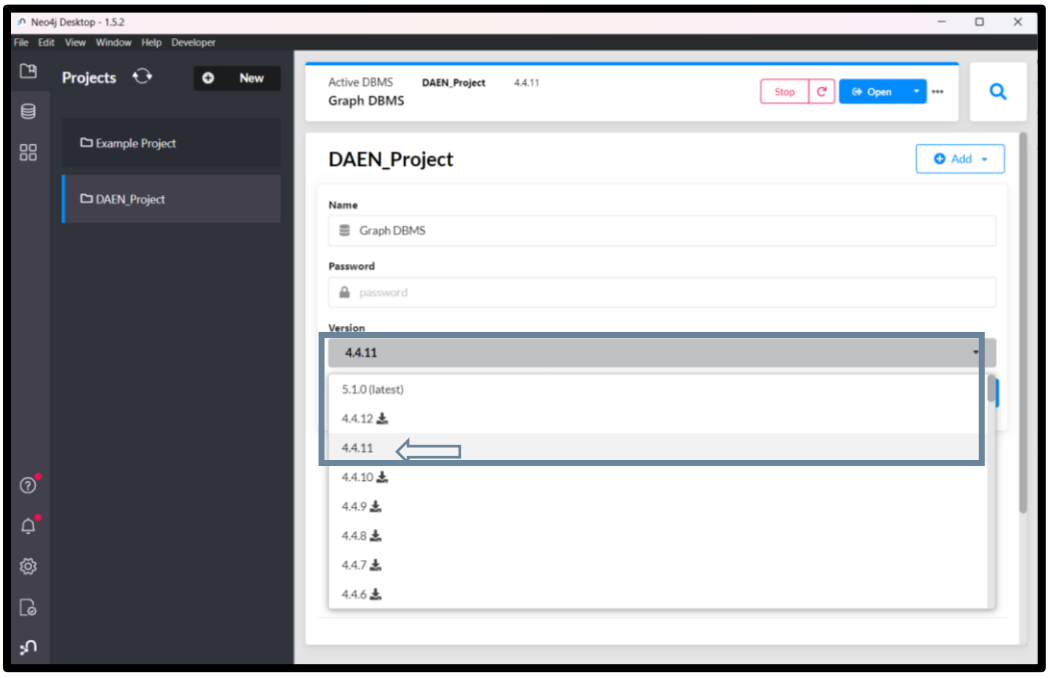


Step 5-

After importing the project, create new DBMS from dumb.

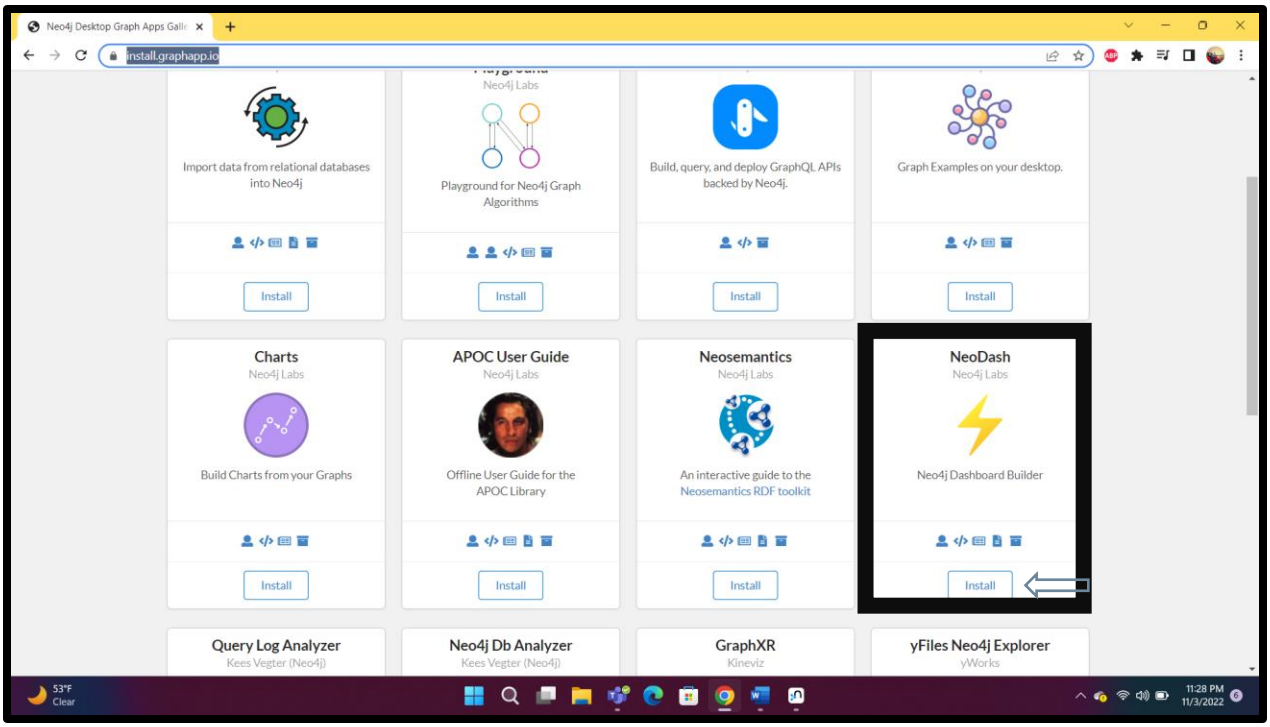


To create DBMS, Give a name and a password then select create. Make sure to change the version to 4.4.11.



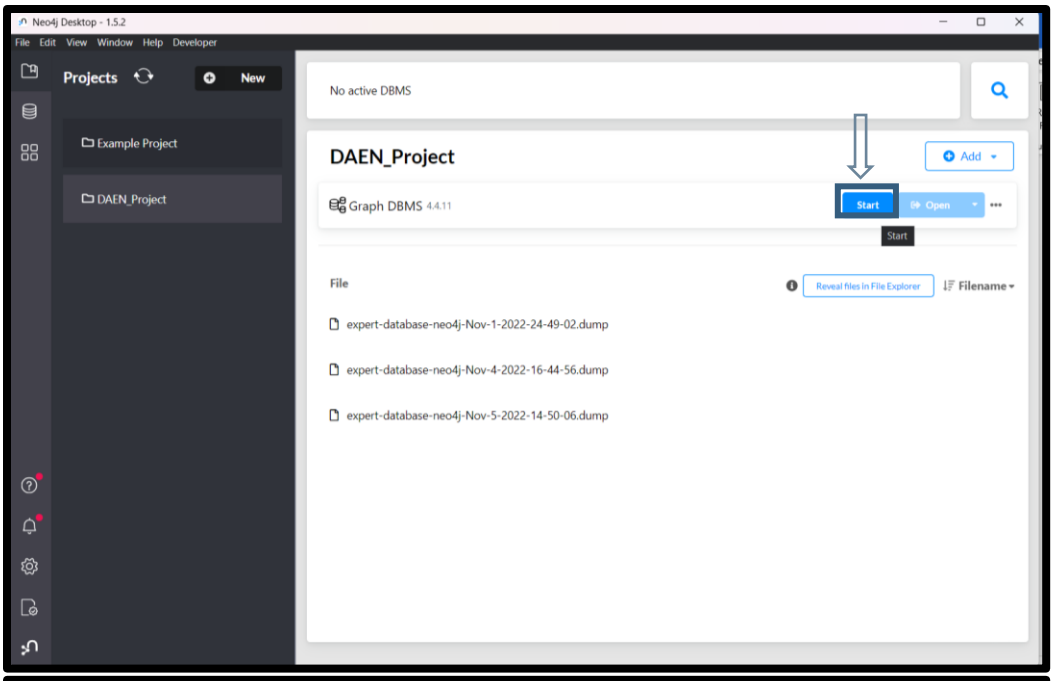
Step 6- Installing NeoDash

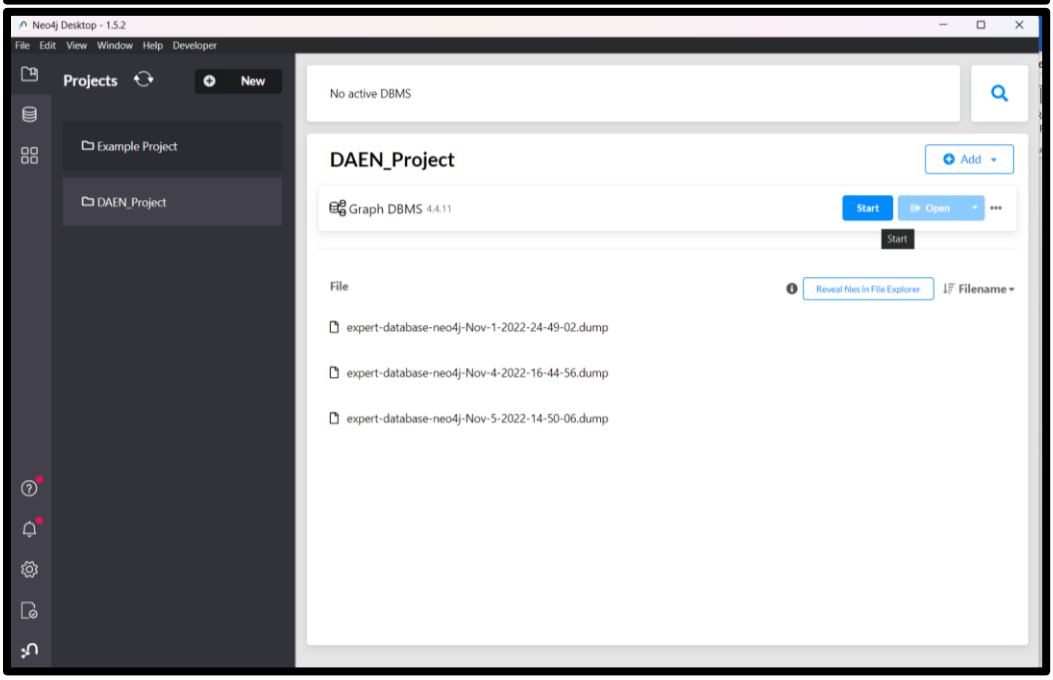
Open https://install.graphapp.io/ on the web browser and select NeoDash.



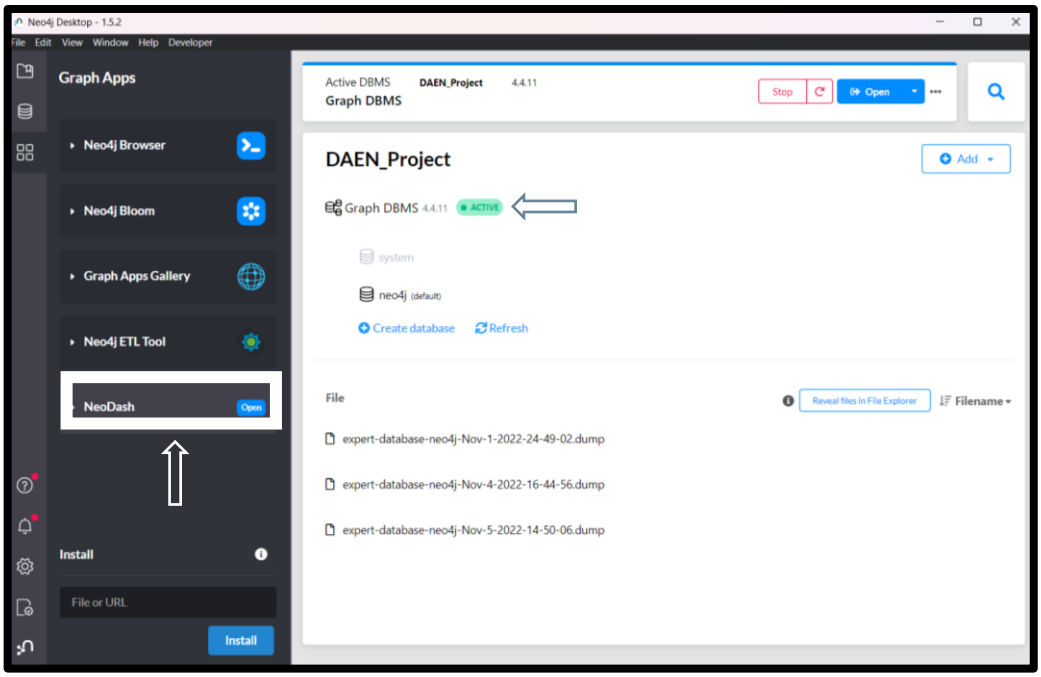
Step 7- Start Database and connect to Neodash

After installing Neodash, Start the database.



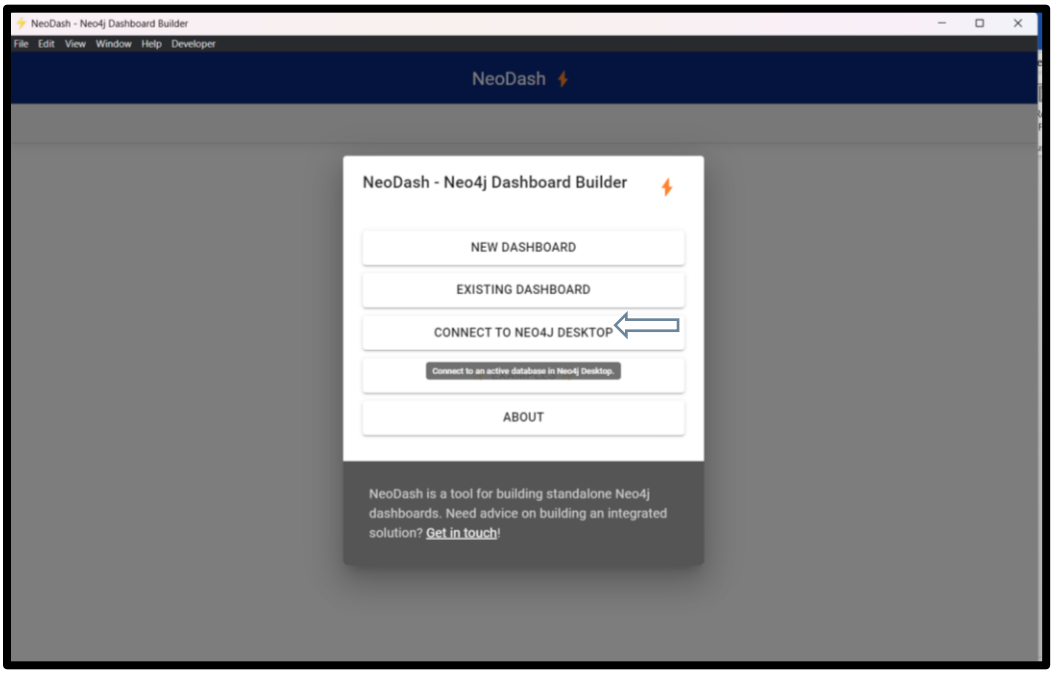


Select Graph apps and open NeoDash.

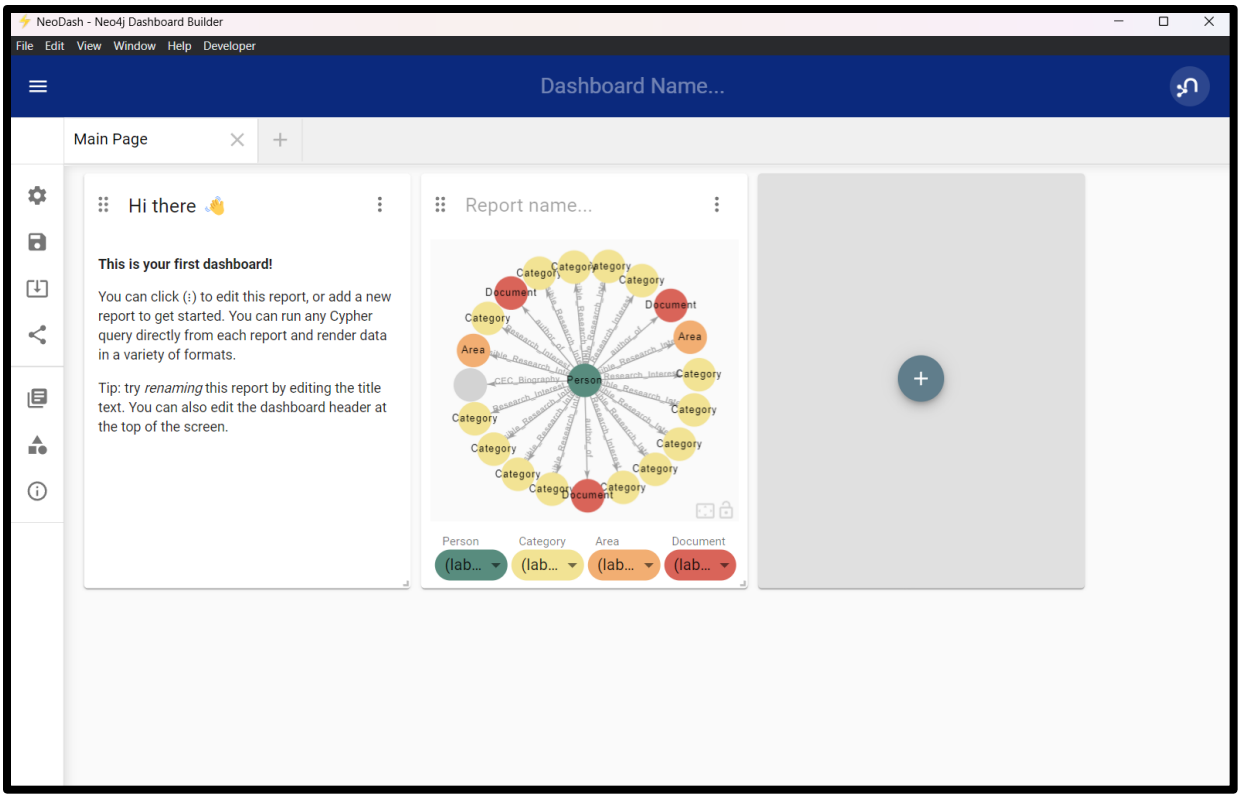


Step 8- Connect Neodash to Neo4j

Make sure that the database is in active Status and then open Neodash.

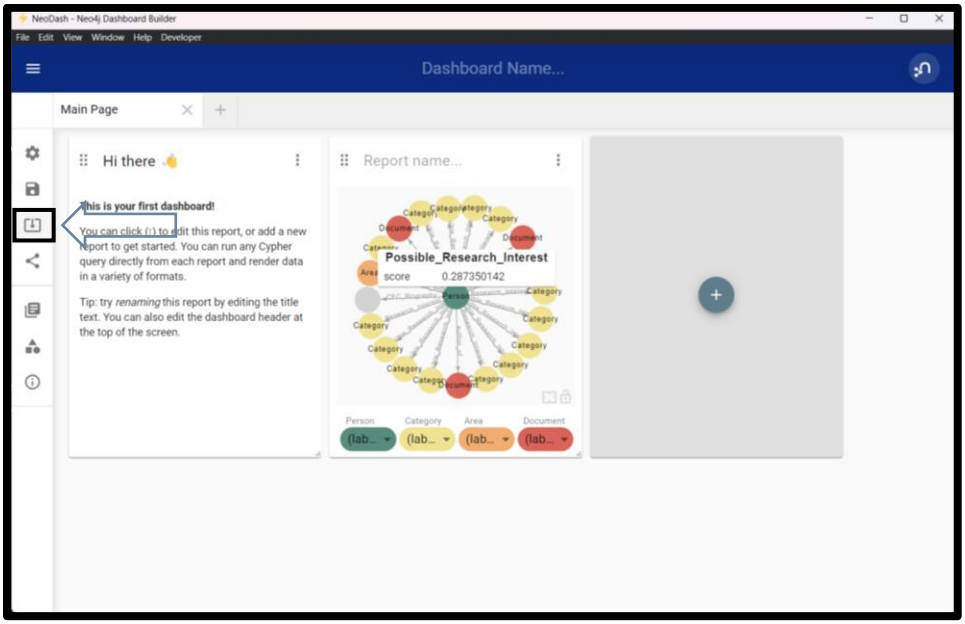


After Connecting the Neo4j Desktop, you should be able to use the dashboard.

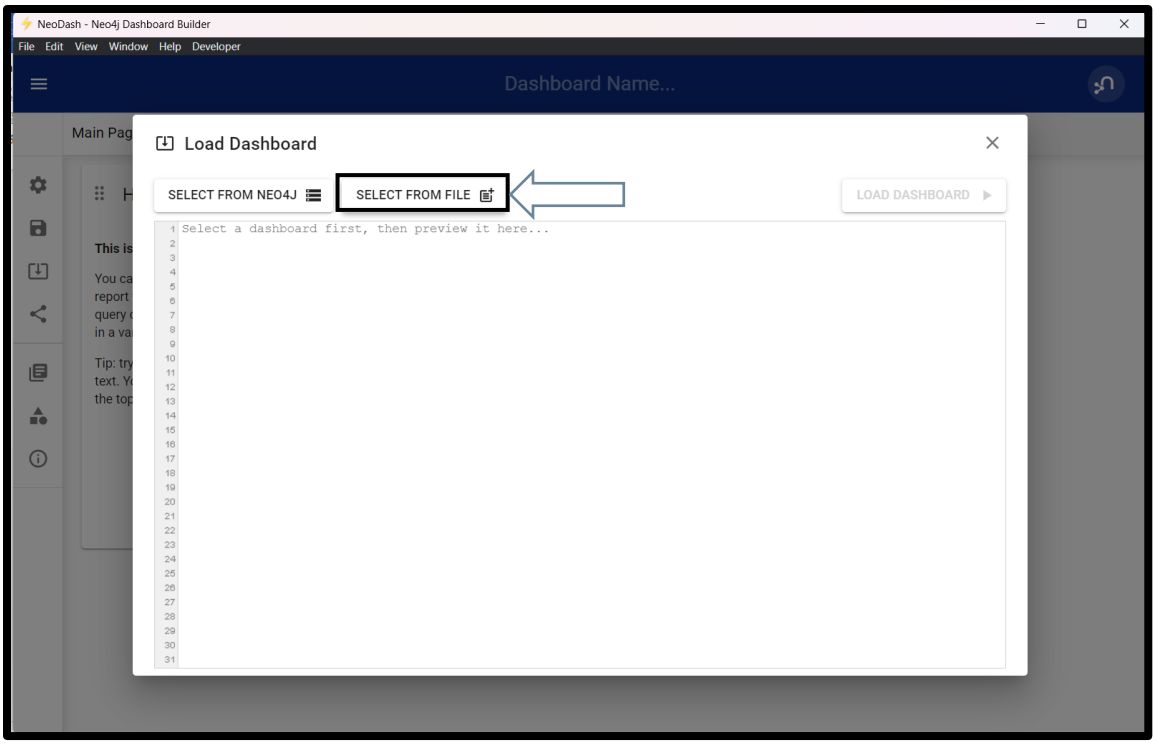


Step 9 – Accessing Dashboard

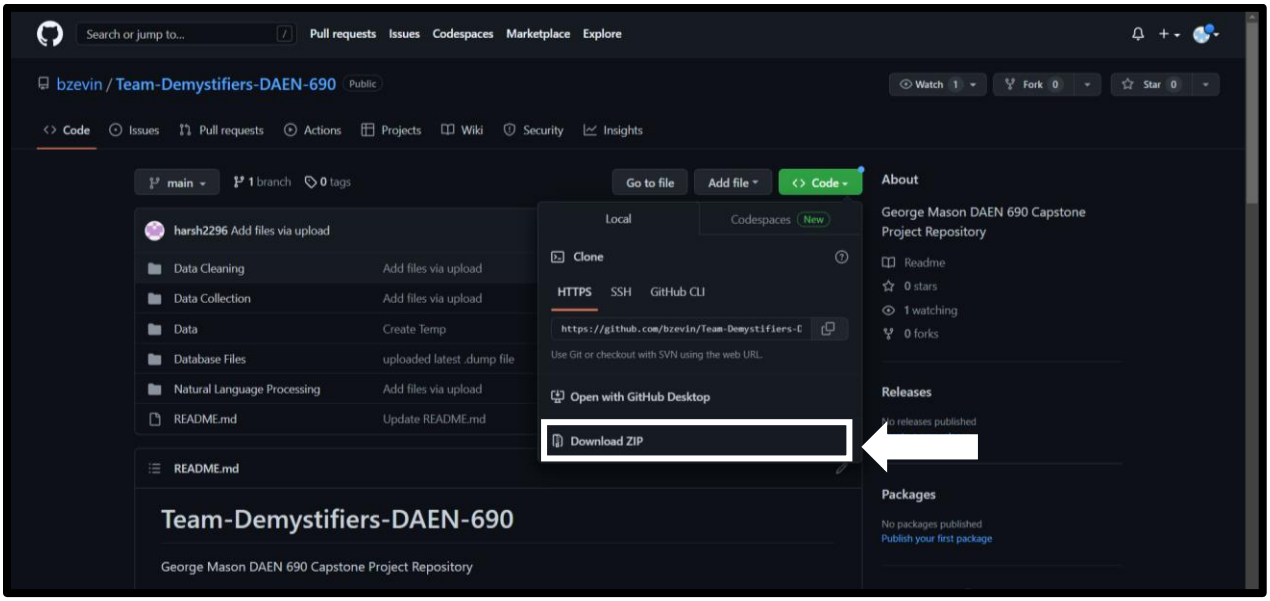
After connecting NeoDash to Neo4j, to access the dashboards upload them into NeoDash. Select upload on the left side of the NeoDash menu.



And then select to upload from file.

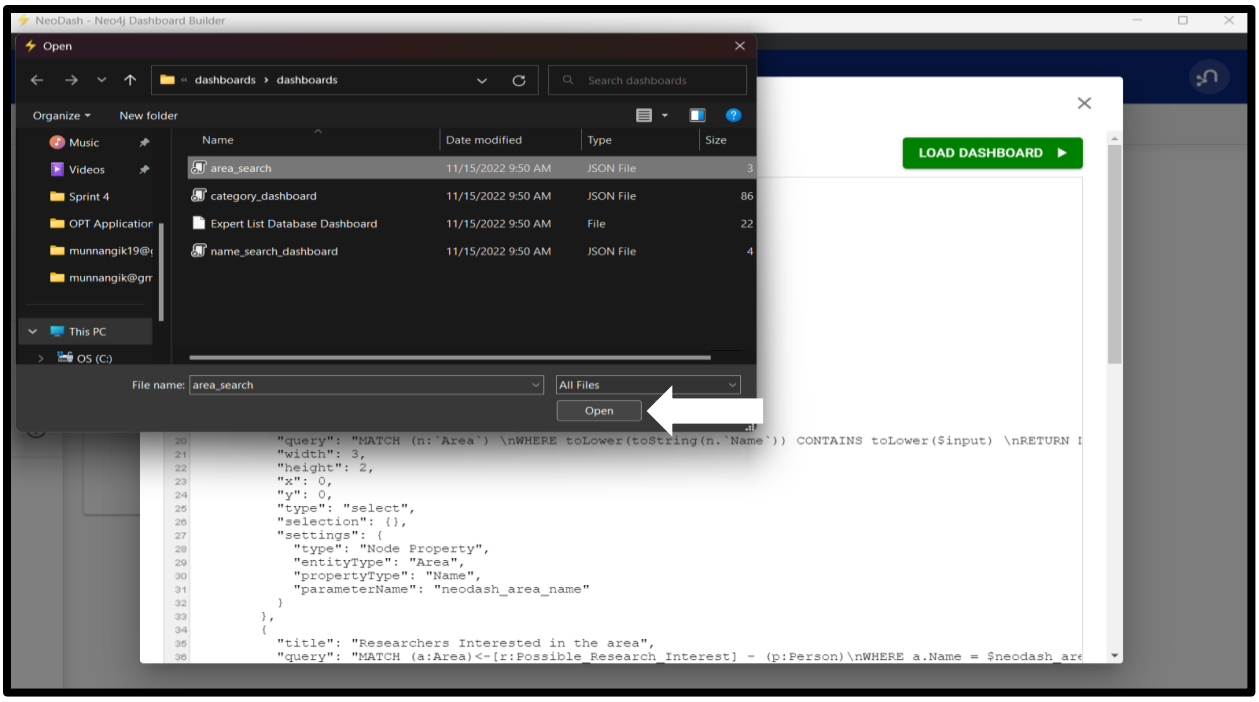


Select the Dashboard from the folder which was downloaded from GitHub. You can find the Dashboard files in the DAEN_Project folder. [27]



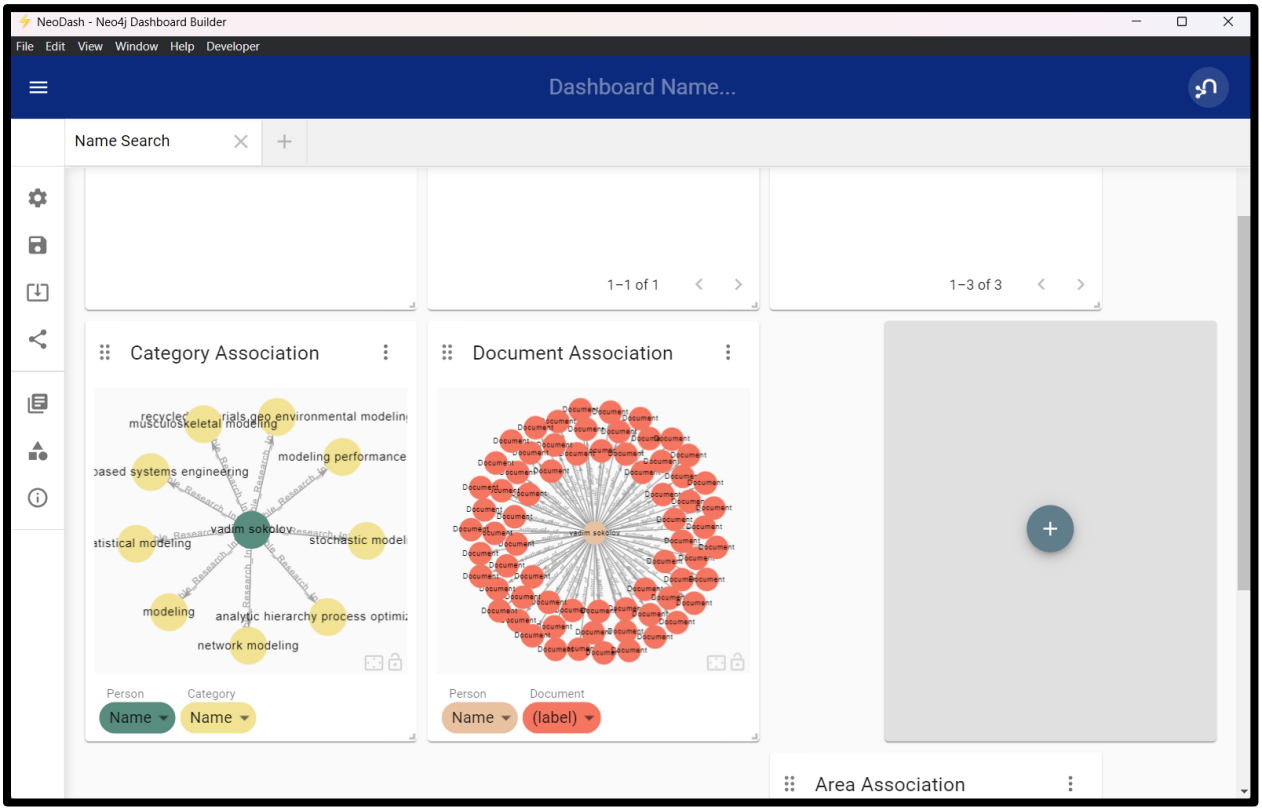
You can follow the following path to access the Dashboard

[Team-Demystifiers-DAEN-690/Database Files/DAEN_690_Project/]



After selecting the dashboard Select Load Dashboard on the Top right side of the Pop-up.

And you should be able to access the dashboard.



10 Appendix C: Risk Section

10.1 Sprint 1 Risks

Risk Name	Description	Probability	Impact	Mitigation
Miscommunication	Team members misunderstand the facts of the task or project	Medium	High	Keep Notes and Review Each others' tasks
Solo Conversation	One person speaks a majority of the time during client meetings	High	Low	Make sure everyone has a talking point
Tunnel Vision	Only focusing on one aspect of a task	Medium	Medium	To ask questions and communicate with the group
Order of Operations	Moving on to another task when a previous task wasn't completed yet	Low	High	Identify what tasks are necessary to complete others
Task Management	Task Assignments being assigned without completing previous tasks	Medium	Low	Focus on Youtrack board during daily scrum session

Throughout the first sprint of this project, our team ran into a variety of different problems. Some of these caused significant issues, and some caused minor issues. We were able to overcome nearly all of them in the first sprint, but some of the repercussions of these risks caused some of the tasks that were supposed to be completed in sprint one spill into sprint two.

The first risk that was identified was that there was some miscommunication between team members and the client. A couple of our team members misunderstood the materials our team had access to and what could be provided to us. This problem caused an incorrect problem space, and that section of sprint one had to be redone in order to describe the problem space accurately. This risk can be mitigated in the future by keeping notes of all meetings within the group and all meetings with the client.

Our team's second problem during sprint 1 was that one or two people in our group would take control of client meetings and do the majority of the speaking. This happened nearly every meeting, but it had a relatively low impact. In order to mitigate the problem in the future, our team needs to ensure that everyone in the group provides input in future client meetings instead of taking a backseat. A way to

make sure this happens is to ensure everyone in the group has some talking points prepared for the meeting.

The third risk that developed during the spring was instances of tunnel vision. Our group members would fixate on one aspect of the assignment instead of looking at the big picture. For example, we focused on old instructions that were not accurate within the scope of our project, and some of us got fixated on writing code and collecting data before we had reached that step yet. The way we mitigated that risk was to ask questions in our team meetings and ensure everyone knew precisely the scope of their task.

The fourth problem during the sprint was not recognizing and following the order of operations that should have been taken during the sprint. Our team did not realize that all of the sections of the report lead into one another. The background leads into the problem space, the problem space leads into the research, and so on. When tasks were created, we only focused on our individual tasks instead of helping to make sure each previous task was completed thoroughly. This caused a significant problem in the research section. Instead of looking for articles and instances of something similar to our project, we just jumped to different aspects of the problem space to research instead of looking at the project as a whole. Due to this error, another possible solution was discovered late in the sprint, and further research needs to be done in sprint two to make a final decision on the path we want to take in the solution space. In order to avoid this risk in the future, we need to identify which tasks are connected and the order in which they need to be completed.

The final risk our team encountered was a lack of task management. Throughout the three weeks we had for sprint one, we laid out tasks each person should complete that week. We discovered that some tasks took longer than a week to complete, and some were frankly forgotten. When the following week's assignments were given out, some people could not complete them due to the incompleteness of their previous assignments, which created a snowball effect. The way to mitigate this task in the future is to focus on the Youtrack board during the daily scrums so everyone can be reminded of what they need to complete that week in order for everyone else to stay on track.

10.2 Sprint 2 Risks

Risk Name	Description	Probability	Impact	Mitigation
Same Name	There could be multiple people with the same name in a database	High	High	Add GMU to the search
Missing Abstracts	Articles in databases could be missing abstracts	Medium	Medium	Find article in another database to see if it has an abstract or parse through article
Access	Two-factor authentication makes it nearly impossible to access some databases	High	High	Manually collect data through API of databases
Unknown Format	Database is designed in unfamiliar format making web scraping difficult	Medium	High	Use Selenium package to open webpage
Incomplete Faculty Lists	Faculty names on CEC website do not reflect all current CEC faculty and researchers	High	High	Collect current CEC faculty and researcher names from other sources and/or CEC POC

Throughout the second sprint, our team identified a few issues that create roadblocks and could even cause us to change our solution space. This sprint focus on data and data collection specifically for us. Due to this, all of the risks identified in this sprint deal directly with data collection.

The first risk our team identified is if there are multiple researchers in a database with the same name. This instance creates an issue because we will be searching through databases using the researcher's name to identify research they are conducting or what they have done in the past. If two people have the same, we could get articles written by a different person with the same name, which would lead to

saying our researcher is an expert in a topic they are not experienced in. This risk has a low probability but has a high impact because if this were to occur, it would lead to incorrect data being added to our database, which could hinder potential users. The way we plan to mitigate this risk is to add a "GMU" tag to the search to only include people with a relationship with George Mason University. The tag will only be included if a previous search shows the risk to be true. The added tag will hopefully remove any people with the same name as our researcher.

The second risk our team identified is if there are missing abstracts about research articles written by a researcher. It is standard practice to have the abstract or description added to the webpage for any research article written that is included in a database. However, we discovered this is not completely true in the GMU library portal. Some of the articles in the database do not include an abstract or description at all. Our team's plan to solve the problem if encountered is to search for the article in another database to see if it is included and has an available abstract. If our team cannot find the article in another database, we will need to go into the research article, manually take the abstract, and add it to the dataset.

The third and fourth risks have similarities in their impact on the solution space. Both of these risks have the potential to remove the possibility of web scraping databases from the solution space. The third risk is that web scrapers could be blocked from accessing the databases our team is focusing on. Web scraping is not the most approved practice, and some companies look down on it because it could potentially steal data and give access to information that requires further access to be granted. Some databases require two-factor authentication, making it extremely difficult to access the desired webpage. Other methods include designing URLs that are not designed in any discernible format, making it extremely difficult to navigate through web pages. If this risk is encountered, our team will have to manually go through the research databases and collect the data using the database API. This option is very time-consuming, which is why this risk has such a high impact.

The fourth risk is if the database format is unknown and unfamiliar to our team. Our team has experience using web scraping with standard HTML web pages. However, some of the databases use a more advanced form of JavaScript that has embedded information in the webpage, which makes the HTML extremely difficult to navigate. Due to the project's time constraint, we cannot get distracted by something that would take our team more time than available to figure out. If we do run into this problem, our team will use the selenium package to open the webpage in a new instance of google chrome add determine if the HTML code is understandable. If the code is, our team will continue to web scrape. If not, our team will take the same actions to mitigate risk three; we will manually search through the database to collect the necessary information on the researchers of our dataset.

The final risk we identified in sprint 2 is if our faculty list is incomplete after collecting the names from the CEC website. The probability and impact of this risk are both high because after researching the CEC website, it appears there are some missing researchers. The names are necessary if our team wants to create a practical prototype. If this becomes an issue, our data is incomplete, and not every faculty or research member is included on the CEC website. We will mitigate this problem by attempting to locate another source for members of the CEC, or we will reach out to an administrator of the CEC to see if our team can receive a complete department list directly.

The risks identified during this sprint show that a solution space is never set in stone, and roadblocks can occur during any step of development. Although luckily, none of these risks make the project impossible, but our team will have to spend more time in the data collection portion than anticipated.

10.3 Sprint 3 Risks

Risk Name	Description	Probability	Impact	Mitigation
Underdeveloped dictionary	The dictionary of NLP is missing keywords	Medium	Medium	Correlate Dictionary with Bag-of-words and research topics in the dictionary to add addition words.
Needless words in Bag-of-Words	Words that don't provide relevant information to the publication remain in the Bag-of-Words	Low	Low Impact	Go throw initial Bag-of-Words and add needless words to a list and remove all of those words.
Name differences	Names for authors are different from source to source	High	High	Standardize names across all source. CEC will be the standard source.
Keyword Multiple Area or no areas	Keywords can be included in different sections of the Dictionary or can be unconnected	Medium	Low	Categories with multiple areas will connect to multiple areas and if added categories without links will stay unlinked.

Throughout the third sprint, our team identified a series of risks that could create problems with identifying the research topics where faculty members are experts. This sprint focuses on the algorithms and techniques used to complete the project. Even though none of these risks changed the

solution space, some of them did cause us to think about Natural Language Processing from a database-friendly perspective instead of a raw data perspective.

The first risk identified was if our dictionary for the basis of database and natural language processing was underdeveloped. If not enough identifying words are established in the dictionary, our team could miss keywords within the abstracts of faculties publications. If we miss keywords in the abstracts, our database could miss links between faculty members and research topics. The impact and probability of this risk are medium because missing links between faculty members and research topics mean our final database is incomplete. Our team mitigated this risk by conducting research on the research topics in the database to add more identifying words to the dictionary. Our team also combined the data collected from NIST to add more identifying words. The NIST data set contains publications published with funding from NIST or in an attempt to gain NIST funding. All of the publications contain a subject and keywords field. Those fields will be added to the dictionary in an attempt to create better links between professors and their research.

The second risk identified is having needless words within our Bag-of-words. The Bag-of-words is designed to remove stop words and any other words deemed unnecessary by the user. The abstracts and descriptions of publications will contain many other words that are not needed within this project's scope. Since these words are unnecessary, they can be removed, decreasing the natural language processing run time. The probability of this risk is high because abstracts are written academically and have many similarities, which are unnecessary for identifying research subjects. The impact of the risk is low because if the unnecessary words are not removed, it will just increase the run time of our natural language processing. This will not affect the database or the front-end tool. Our team mitigated this risk by counting the frequency of each unique word and removing the words with the highest counts that are unrelated to a research subject. This process will have to be conducted manually.

The third risk identified is that the data collected from the sources have different formats when labeling the authors. For example, some Google Scholar profiles use nicknames made by the profile owner. These nicknames are not included within the data collected from the CEC website, LinkedIn, and Academic search complete only contains the author's last name and the first letter of the first name. The probability and impact of this risk are both high because if the names between sources are not standardized and our team does not correct them, some faculty members conducting research will appear not to be conducting any research since none of the publications are connected to their names in the CEC dataset. In addition, we will not be able to create likes between people and research topics if the names are not standardized. Our team mitigated the risk by manually collecting the names that were not identical and standardized them using the CEC data.

The fourth risk identified is that some categories within the dictionary belong to multiple areas. In contrast, most of the other categories added to the original dictionary do not belong or have been assigned a particular area. This is a potential risk because some faculty members could be assigned to a category for an area that they are not familiar with. The probability of this risk is high because it was identified very quickly, and there are a couple of examples of this occurring. Due to time restraints, any other categories added to the report will not be assigned to a certain dictionary area. The impact of this risk is low because most categories belonging to multiple areas are extremely similar, and the database user will know the meanings of the categories and areas and be able to differentiate between them. As for the categories that are added and do not belong to any specific area, the category name should be

enough information for the user to recognize the meaning of the category, making the connection to a certain area unnecessary.

The risks identified during this sprint show that a solution space is never set in stone, and roadblocks can occur during any step of development. Although luckily, none of these risks made the project impossible, but our team had to spend more time than anticipated formatting the data to complete Natural Language Processing and graph database development.

10.4 Sprint 4 Risks

Risk Name	Description	Probability	Impact	Mitigation
Under-Developed Dictionary	Categories contain many different phrases for the same concept.	High	Medium	Introduce way to resolve problem into future work section.
Errors In Database	Errors in person information and document information.	Medium	Low	Manually fix mistakes in CSV files and reload database.
Case-Sensitive Relationship and Node Labels	When trying to access a field in a relationship and/or node, the field is case-sensitive.	Low	High	Add a database schema graph to the first dashboard.

Throughout the fourth sprint, our team identified a few risks that could create problems deploying our front-end tool for this project. This sprint focuses on the visualizations developed to display our findings or the development of front-end tools to display our results. Even though none of these risks changed the solution space, some cannot be addressed at this project stage and will be addressed in the future work section.

The first risk that was identified was that our dictionary was underdeveloped. After adding more categories to the dictionary, we discovered that many of the additions were multiple phrases for the same concept. Additionally, our team did not place the newly added categories into proper areas to create a more refined structure. The impact of this risk is medium because it does not break the database or front-end tool. It makes the database unnecessarily messy and separates possible links with one refined category into links to multiple phrases with the same meaning. This causes users to search for multiple phrases to find accurate information. The probability of this risk is high because it is an existing problem in our design, and due to the time when the problem was discovered, there needed to

be more time left in the project to fix it correctly. Because of this, our team decided to mitigate the risk by introducing a way to resolve the problem in the future work section of the report.

The second risk is that errors were discovered in the database due to cleaning mistakes in the CSV files used to load the data. For example, faculty member information was incorrect because some rows did not split properly, and some document files had columns containing duplicate information. The probability of this risk is medium because the errors were infrequent, and the impact of the risk is low because the data loaded into the database was not incorrect. It either had duplicate entries, so there would be multiple instances of the same object, or a person's email address was not in the proper format. All of the data was still usable and did not disrupt the database. The mitigation to this risk was that our team manually went through the CSV used to create the database, found and fixed the errors, and then reloaded the database with the changed files.

The third and final risk discovered was that when developing NeoDash dashboards, the cypher query language is case-sensitive. Therefore, a developer must be specific about relationship and object names. If a user wanted to create a new dashboard to display the data differently, they would have to know the exact names of all the relationships and nodes. The probability of this risk was low; this risk would only occur if a user wanted to create new visualizations or additional dashboards. Our front-end tool is designed so additions would not be necessary, and a user should be able to obtain all the information they want using the existing dashboards. However, the impact of this risk is high because if someone did want to create new visualizations, it would be challenging to do so without a thorough understanding of the database structure. Our team mitigated this risk by adding a database schema graph to the first dashboard containing all of the relationship and node names. A developer can use this schema to create any new visualizations they would like.

10.5 Sprint 5 Risks

Risk Name	Description	Probability	Impact	Mitigation
Presentation Timing	Going over or under the allotted time for our presentation.	Low	High	Practice the presentation
Presentation Errors	Accidentally giving false information during the presentation	Low	High	Practice the presentation in order to catch mistakes

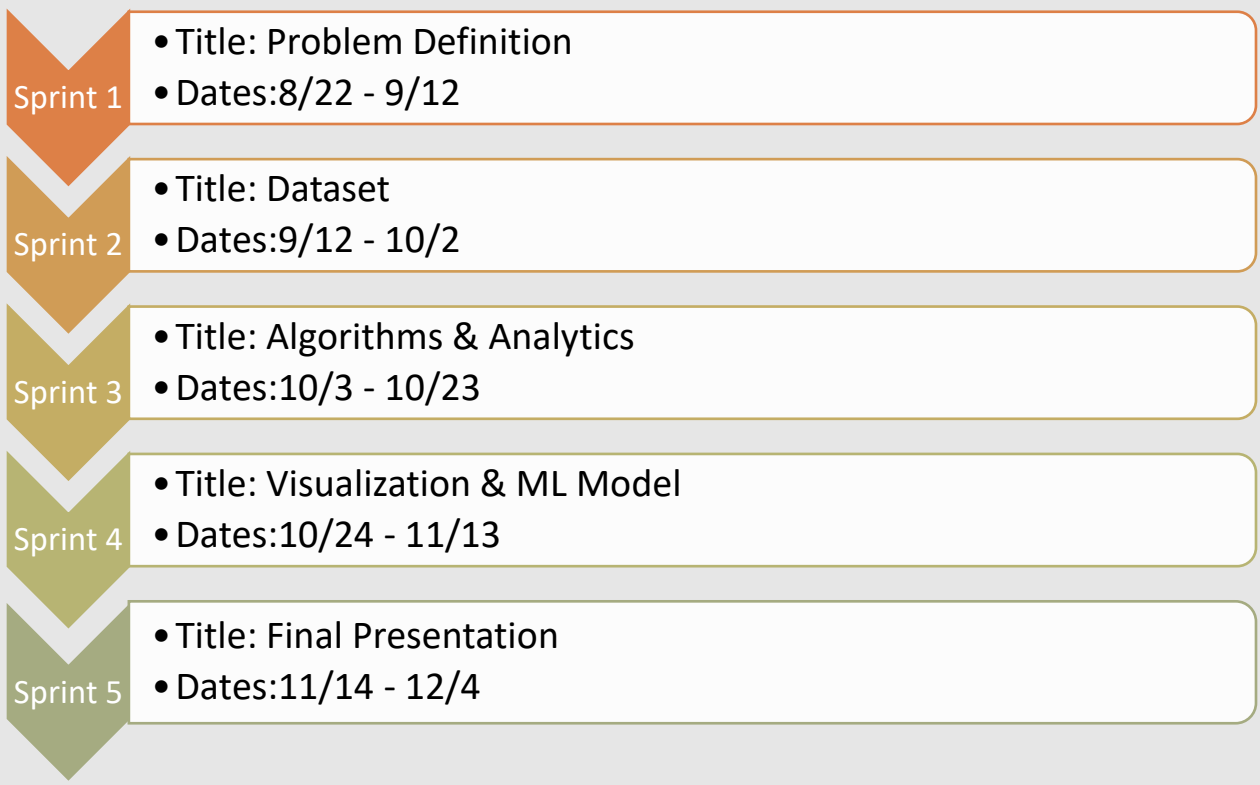
Throughout the fifth and final sprint, our team identified a few potential risks involving the final presentation. This sprint focus on the findings, summary, and future work of the report as well as the final presentation of our project. The report has no risks, just the final showcase presentation.

The first risk identified was the presentation timing. Our team had to complete our presentation with a minimum time of 20 minutes and a maximum time of 25 minutes. If our team completes our presentation in under 20 minutes, we will lose points on our final grade. If our team does not complete the presentation in under 25 minutes, we will be stopped by the instructor before we have finished, and we will lose points on our final grade. To mitigate this risk, our team practiced the presentation multiple times and obtained an understanding of the amount of time required to complete our presentation. The probability of this risk is low because our team practiced the presentation, and the potential impact is high because we could lose many points in our final grade.

The second risk identified was potential presentation errors. Accidentally giving false information during our presentation is a significant issue because we do not want to misrepresent what we did during this project. We want to give the viewers accurate information if they want to use our product. The probability of this risk is low because of our mitigation, which was to practice the presentation multiple times and adjust slides and scripts accordingly to remove any mistakes. However, the potential impact of this risk is high because if we provide any false information, the viewers might misunderstand the project's purpose or one of our project's vital steps.

11 Appendix D: Agile Development

11.1 Scrum Methodology



11.2 Sprint 1 Analysis

In Sprint 1, our team spent considerable time understanding what Dr. Wells wanted out of this project, the motivation behind the project, and what approaches we could take to address the problem. After our meetings with Dr. Wells, we felt we had a good grasp on the components needed to address the problem appropriately, which became our User Stories. However, our background research was initially restricted to the methods we wanted to implement to address the problem based on our domain knowledge; this became problematic because our background research never accounted for previous research.

Our team sought out previous research articles related to the issue we are trying to address, and we found an article written by researchers at the University of Pittsburgh. Their approach used graph databases, which is an approach we never considered prior to finding the article. After finding this article, the direction we wanted to take for the project became a lot clearer. Our last conversation with Dr. Wells in Sprint 1 was our most productive meeting thus far. We were able to both discuss the approach using graph databases and learn how Dr. Wells' educational background tied into the birth of this project.

Our team meets daily for 15 to 30 minutes. Our team had some trouble with writing different components of Section 1 of the report; however, after receiving feedback from Professor Schmidt, we understood the expectations of the sections and subsections of Section 1. With YouTrack, managing activities was very organized. However, our team could do a better job of figuring out how to assign different reviewers for different tasks. Communication of expectations and tasks between teammates has been satisfactory but could use improvement.

11.3 Sprint 2 Analysis

In Sprint 2, our team spent time learning about the data, namely how to acquire the data and what hurdles we would have to overcome to acquire the data. We recognized that one of the biggest challenges for this project was acquiring data because—unlike a lot of projects—we are not given a dataset from the customer. That said, team communication and organization became even more important for this part of the project to move proactively. We identified potential challenges, addressed them to Dr. Wells, and made sure to document them in the report immediately. For example, we found that there were ethical concerns related to scraping data from LinkedIn. This discovery slightly changed our approach to web scraping and gave us material to write about in the final report. If we had identified this and handled it much later, our timeline would have shifted significantly.

Our data assessment also found that trying to automate web scraping for Academic Search Complete was difficult because it not only required a GMU account and multi-factor authentication to access. We identified this issue early and came to the conclusion that a brute force method of searching (and saving the results of the queries in .csv files) over 200 faculty and researchers in the CEC was our best approach to obtaining those data. With the more proactive approach to data assessment in Sprint 2, we have confidence that we can spend more time working on the algorithms we will be using in Sprint 3.

Our team continues to meet daily for 15 to 30 minutes. We are consistently getting better at identifying and communicating risks to each other so that if someone needs help, others can offer a helping hand.

One area of improvement we could work on is staying on top of adding exit criteria to YouTrack tickets. Commonly, we create tickets outlining the task(s) that need to be done, but sometimes forget to indicate what criteria need to be met for the ticket to be considered "done." Organization and communication are always elements within a team that benefit from consistent improvement.

11.4 Sprint 3 Analysis

In Sprint 3, our team spent considerable time scraping, formatting, consolidating, and learning how to store data. We recognized that the greatest challenges in this sprint were formatting and learning how to store data. Furthermore, one of our data sources had to be collected by brute force due to our inability to scrape data past multi-factor authentication. Different data sources stored data in different formats, so we had to spend time deciding what information we wanted for our graph database and how we wanted our data to be organized.

Less time was spent documenting progress and contributing to the report in this sprint compared to other sprints. Hence, our 33% and 66% interim status updates within our report were lighter in content than previous sprint efforts. However, the majority of this sprint was focused on the development of Python scripts to gather, organize, and store data. There were times when managing activities during this sprint became difficult. With the number of data sources we were using, there were moments of miscommunication related to keeping track of where different datasets were stored and what kind of analysis was being done on a dataset. These moments of miscommunication served as a useful way for us to understand we may need to employ better organization in our file structure

Organization is of utmost importance because in Sprint 3, responsibility of a dataset shifted from an individual researching and writing about a single dataset in the final report to developing scripts to consolidate each individual's datasets. We learned that organization is important when it comes to merging everyone's individual work into one dataset. We identified that we not only need to organize our GitHub repository, but also spend time in the near future updating Sections 1 and 2 of the report with discoveries we had made in Sprint 3 (e.g., choosing NeoDash as our front-end GUI option instead of Microsoft Power BI or Tableau). Conducting maintenance would mitigate any confusion related to where datasets and scripts are stored in our shared repository.

11.5 Sprint 4 Analysis

In Sprint 4, our team spent most of the time creating dashboards for Neo4j and creating histograms and bar charts to understand the distribution of top 20 areas, top 20 categories, and cosine similarity scores across all areas & categories in our database. Compared to other sprint efforts, this sprint was the least intensive because the application we decided to use was easy to interact with. The most challenging part of this project was in Sprint 3, when we gathered and stored data collected from online sources.

We also began a lot of project maintenance. This included updating previous sections of the project report to reflect work we had done in sprint 3, reorganize our GitHub repository, and write an instruction manual on how to use Neo4j Desktop and NeoDash (both as another appendix in this report and on the GitHub repo README). The additional time available in this sprint allowed us to look back at the report and reassess what we had written about before. For example, we removed some content related to front-end applications we considered using before choosing NeoDash. Reevaluating previous sections of the report will continue into Sprint 5.

11.6 Sprint 5 Analysis

In Sprint 5, our team spent most of the time working on the final showcase presentation slide deck. Initially, our tasks were designed so that everyone would focus on their particular section of the presentation. In hindsight, this approach could have used more thought. During our first dress rehearsal, it was brought to our attention that our presentation was rich in content but was disjointed and was missing information related to why the project was important and why the audience should care. These comments were accounted for in the following week. After multiple rehearsals between the in-class dress rehearsals, we believe that our presentation has become smoother and more connected.

Our team made these adjustments quickly and efficiently, but what has yet to be seen is how our newest developments to the presentation play out in person. Our rehearsals were limited to Microsoft Teams because of scheduling and location conflicts over Thanksgiving weekend. So, important presentation features, such as body language and engaging with the audience, were not in practice between in-class dress rehearsals.

12 References

[1] "Center of Excellence in Command, Control, Communications, Computing, Intelligence, And Cyber," George Mason University, 2022. [Online]. Available: https://c4i.gmu.edu/. [Accessed 8 September 2022].

[2] B. Rahdari and P. Brusilovsky, "Building a Knowledge Graph for Recommending Experts," *DI2KG@KDD,* 2019.

[3] IBM Cloud Education, "Knowledge Graph," IBM, 12 April 2021. [Online]. Available: https://www.ibm.com/cloud/learn/knowledge-graph. [Accessed 6 September 2022].

[4] "What Is a Graph Database?," [Online]. Available: https://neo4j.com/developer/graph-database/. [Accessed 12 Sept 2022].

[5] K. Mehta, M. Salvi, R. Dand, V. Makharia and P. Natu, "A Comparative Study of Various Approaches to Adaptive Web Scraping," in *ICDSMLA 2019*, Singapore, 2020.

[6] "ParseHub | Free web scraping - The most powerful web scraper," [Online]. Available: https://www.parsehub.com/. [Accessed 31 08 2022].

[7] "urllib — URL handling modules — Python 3.10.6 documentation," [Online]. Available: https://docs.python.org/3/library/urllib.html. [Accessed 31 08 2022].

[8] "Requests: HTTP for Humans™ — Requests 2.28.1 documentation," [Online]. Available: https://requests.readthedocs.io/en/latest/. [Accessed 31 8 2022].

[9] "Beautiful Soup Documentation — Beautiful Soup 4.9.0 documentation," [Online]. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/. [Accessed 31 8 2022].

[10] J. Axenbeck and J. Kinne, "Web Mining of Firm Websites: A Framework for Web Scraping and a Pilot Study for Germany," *SSRN Scholarly Paper,* 2018.

[11] IBM Cloud Education, "Natural Language Processing (NLP)," 2 July 2020. [Online]. Available: https://www.ibm.com/cloud/learn/natural-language-processing.

[12] "Knowledge graph," Wikipedia, 3 10 2022. [Online]. Available: https://en.wikipedia.org/wiki/Knowledge_graph. [Accessed 31 10 2022].

[13] "Best Graph Databases," [Online]. Available: https://www.g2.com/categories/graph-databases. [Accessed 12 Sept 2022].

[14] "Graph Database vs. Relational Database: Key Differences. SearchDataManagement,," [Online]. Available: https://www.techtarget.com/searchdatamanagement/feature/Graph-database-vs-relational-database-Key-differences.. [Accessed 12 Sept 2022].

[15] "What Is a Graph Database," [Online]. Available: https://www.oracle.com/in/autonomous-database/what-is-graph-database/.. [Accessed 12 Sept 2022].

[16] "Why Neo4j? Top Ten Reasons," [Online]. Available: Neo4j Graph Data Platform, https://neo4j.com/top-ten-reasons/. [Accessed 12 Sept 2022].

[17] "Neo4j Graph Data Platform," Neo4j Inc., [Online]. Available: https://neo4j.com/. [Accessed 08 September 2022].

[18] "Neo4j Graph Data Platform," [Online]. Available: https://neo4j.com/developer-blog/neodash-2-0-a-brand-new-way-to-visualize-neo4j/. [Accessed 27 Oct 2022].

[19] "NeoDash - Dashboard Builder for Neo4j," [Online]. Available: https://neo4j.com/labs/neodash/.

[20] H. N. Rozear, "Where Google Scholar stands on art: an evaluation of content coverage in online databases," *Art Libraries Journal,* vol. 34, no. 2, pp. 21-25, 2009.

[21] "Google Scholar Search Help," [Online]. Available: https://scholar.google.com/intl/us/scholar/help.html#coverage. [Accessed 18 09 2022].

[22] G. C. L. MAdian Khabsa, "The Number of Scholarly Documents on the Public Web," *PLOS ONE,* vol. 9, no. 5, p. e93949, 2014.

[23] "NLTK: A Beginners Hands-on Guide to Natural Language Processing.," 17 July 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/07/nltk-a-beginners-hands-on-guide-to-natural-language-processing/. [Accessed 2022 10 21].

[24] N. G. D. Platform, "15 Tools for Visualizing Your Neo4j Graph Database.," [Online]. Available: https://neo4j.com/developer-blog/15-tools-for-visualizing-your-neo4j-graph-database/.

[25] D. V. Comparison., "Microsoft Power BI vs Neo4j," [Online]. Available: https://www.slintel.com/tech/data-visualization/www.slintel.com/tech/data-visualization/microsoftpowerbi-vs-neo4j. [Accessed 28 Oct 2022].

[26] N. G. D. Platform, "Download Neo4j Desktop," [Online]. Available: https://neo4j.com/download/.

[27] Team-Demystifiers-DAEN-690, "GitHub Repository," [Online]. Available: https://github.com/bzevin/Team-Demystifiers-DAEN-690.

[28] "Publications," www.nist.gov, [Online]. Available: https://www.nist.gov/publications. [Accessed 2 September 2022].

[29] "NSF Award Search: Simple Search.," [Online]. Available: https://nsf.gov/awardsearch/. [Accessed 2 September 2022].

[30] C. C. C. I. a. C. Center of Excellence in Command, "C4I and Cyber Center Attributes," George Mason Univeristy, 2022. [Online]. Available: https://c4i.gmu.edu/c4i-center-attributes/. [Accessed 8 9 2022].