ECM3401: Individual Literature Review and Project

To determine the relationship between each gender's representation in a field and the number of academic paper citations for that field and possibly plot it evolution over time.

Thomas Sturgeon

Abstract

This project aims to develop an application for the visualisation of gender representation in academic papers, to simulate an academic field to uncover trends, disparities, and potential bias within scholarly literature. Using data collection and graph visualisation methods we analysed gender ratios among authors across academic fields and explored trends.

Our results revealed disparities in gender representation, with variation observed between the same academic field but different periods. In contrast, some fields showed a more balanced gender representation compared to others.

Through discussions of our findings, we identified potential factors contributing to these disparities, including cultural bias, institutional policies, and economic inequality. Additionally, we acknowledge limitations in our analysis method, such as challenges in accurately predicting gender based on the author's name alone.

In conclusion, this project provides insights into gender dynamics within academia and talks about further research that could be done to explore this topic with improvements that could be made. However, improvements can be made to allow more insights into how gender representation has evolved.

I certify that all material in this dissertation which is not my own work has been identified.

Contents

1 – Introduction

This project designs and implements a system that aims to represent an academic field through documented reports in academic papers. This will be done by producing a visual representation of the academic field, creating a network graph with individual nodes being the academic papers and the edges being the citations. The node colour will indicate the gender of the author(s) in the by building a network of papers one can gain an overview of gender representation for the field and see at a glance the relative representation. This will be a larger help compared to a list of authors since it is possible that when analysing the data an incorrect conclusion about the authors' gender is created. However, when looking at a graph of colours it is easier to analyse with a quick look since there is already an established difference. This will help to keep the number of human errors lower compared to estimating the gender of uncommon names.

Gender equality and representation is already a well-established research topic in academia with one of the first and most influential papers published being “The Second Sex” by Simone de Beauvior (Beauvior, 1949). This book was one of the beginning studies that is a criticism of patriarchy and sexism. It was an inspiration for women to assert their right to gender justice and equality. Over the coming decades, more and more studies were completed from, a call to arms against the oppression of gender, to more recent studies on unconscious bias towards gender disparities (Marszalek, et al., 2020). These studies show the development of gender representation and how it has changed from the beginning where women are seen as less than men to modern studies where we consciously see the main two genders as the came but unconsciously have a bias towards one.

The reason why it is important to look back on the history of gender inequality and representation is because gender has an important role in our society. It can be about basic reasons such as promoting equality and social justice. This is because we cannot say we have a just and fair planet when people are being treated less than others for no reason. The history of the women’s rights movement starts in 1848 but it has been a long and hard fight for the barest rights (Eisenberg & Ruthsdotter, 1998). Another reason why gender equality is important is that helps enhance economic development, by understanding gender analysis helps understand the implications of gender inequalities. Addressing these inequalities and recognising the contributions of everyone to economic development helps others feel included and helps bring down barriers making economies more inclusive and sustainable. This can be seen in “The Power of the Pill” which was a study focusing on how women's contraceptives affect women’s careers which in turn helps economic development (Goldin & Katz, 2000).

In this report, there will be a brief introduction of what the project aims to achieve and an introduction to what has already happened in this area of study. The design of the project will be described in section 3 detailing the process used in the implementation of this project followed by an account of the developmental procedure with a history of the implementation. Following will be an analysis of the results and my testing of the project.

2 – Project Specification

The overall aim of this project is to identify gender representation within academia and where possible examine the change in that representation over time. This will be achieved by:

- Creating a network of academic papers, references, and citations.
- Developing a model to reliably ascribe a gender to the authors of the network papers.
- Visually differentiate the produced network of academic papers, not only by gender but also the date of publication (To determine any change in gender representation over time)
- Consolidate the output into a graphical method to show any results produced.
- Analyse the trends shown by the produced graphs.

2.1 – Requirements

- Data Gathering

To plot the network, we need to be able to collect data about the academic papers that will be used to create the graph. There are many methods to do this ranging from existing datasets, API and Scraping websites that function as distribution centres for academic papers (e.g. Google Scholar). Later on, this report will further explain the choices of why the method that had been chosen, was used.

- Interface

  This will allow the user to change how the data is being presented. It will allow the user to change things like the starting academic paper, the depth of the network (the number of levels to investigate) as well as the layout of the graph. This report will go on the explain the evolution of the interface and why each component of it is necessary.

- Data Analysis

  This will be the section on where data is manipulated and how the user wants the data to be used. This will be things like predicting the gender of the authors and creating the network that will be plotted.

- Storage

  Once the network has been created the nodes will need to be stored. This will go on the explain whether or not this project chooses to use methods like cloud storage for the data or graphs generated or whether or not each time at runtime the network is re-created, and the data is re-gathered.

3 – Project Design

3.1 – Network Analysis

Network analysis is the study of a network. It examines the relationship between entities for example people, organisations, or documents. In this project, it is the relationship between documents which represent a person or a group of people. Network analysis is used in many tasks today. It helps with the like of the structure of relationships in social networks (Great Learning, 2023). Network analysis can be used because it simplifies the complexity of relationships between a large group of academic papers into an easy-to-understand graph, it is simple to find the exact relationship between two points. It would take more time and resources to find the same relationship when looking at the raw data which is the individual papers. With the data displayed visually, it may also help us to discover whether there are behavioural (by field, the relevance of a paper) patterns that are repeated and how they change over time. It also makes it simple to change the search terms. This project focuses on gender representation however it would be straightforward to alter the program to extract data points like the most commonly cited author in an academic field or the most commonly published year.

Node and Edge Definition

In this project, it is important to be able to understand the network graph that is created. If the user is constantly reading the graph incorrectly then that means any analysis that is extrapolated from it has the incorrect conclusions.

Each node represents a single academic paper. It is then possible to extrapolate the gender of the author of the academic paper and ascribe the gender to the colour of each node. If all goes well then it should assign a colour between blue and red with blue being male and red being female. The reason why the nodes are mapped between the colours instead of being either or is because one paper can have multiple authors of different genders. Many papers are co-written between multiple authors, so it has decided to be a blend of colours on a spectrum depending on the number of authors of each gender varying from dark blue to red.

Each edge represents a citation between academic papers. Using this a plot can be created out of the links representing an academic field and that of related papers. This allows one to paint a picture of gender representation in any academic field by changing the starting paper. Allowing for a large versatility of the application.

3.2 – Data Collection

Academic Paper Collection
This section of the project forms the main backbone of the structure. Without robust data, the project is destined to fail. The first thing achieved by the project and therefore the first task was to find a reliable source of academic paper data. This was not straightforward. Multiple methods were attempted to collect the data with a brief overview being discussed here and a further detailed report in discussing the development. However here I will talk about the final methodology used. Most of the data is being taken from the Semantic Scholar API and a publicly available Python module to easily access the API and extract data.

Attribution Extraction
With the academic papers retrieved from the Semantic Scholar API the project then extracted the data used to form the graph. The data fields extracted were the academic paper title, paper Id, author Id, publication year, paper abstract, academic paper URL and a list of papers referenced Ids. With the Semantic Scholar API, the paper references are used. This is talking about the papers that are cited in the original article rather than all citations of the original article. E.g. all the referenced papers were released before the original article. All the citations were released after the original article.

With the use of the author Id, one then makes another request to the API to collect the information about the author in an attempt to collect the author's first name if only the author Id is available.

Data Manipulation
Data manipulation is required for this project to assign a gender to the author(s) of each paper. The data collection methods that fall under data manipulation is to create data that can be extrapolated from the data collected. This means that the data is subjective and could be incorrect since it is unable to be confirmed. This is mainly concerned with the genders since it is unable to be retrieved from the API. An external source will need to be found to predict the gender of the authors from the author's name. This can be more challenging than expected for several reasons since not all paper has an author associated with it. This is more common with older papers but can happen since not all authors have their names easily accessible. It is more common for the name to be in the format of an initial and last name (e.g. T Sturgeon). This creates problems because it cannot accurately predict the author's gender.

3.3 – Visualisation

Interface
This section describes how the user interacts with the application. The main aspect of the interface is the selection of an academic paper. The interface is created such that the user can enter an academic paper DOI (Digital Object Identifier), or their Corpus Id which is the unique identifier for Semantic Scholar. The DOI was chosen to use since it is accessible from any location where the academic papers could be found. Whereas if the user chooses one of the other Id the user would have to find the starting paper from the Semantic Scholar website.

Network Display
This section describes how the network will be displayed. The network will be displayed on the main section of the page where it will be created surrounding the central node with each node coloured depending on their predicted gender. If the gender cannot be discovered the node will be coloured grey. Once the graph has been plotted this allows for a person to analyse the network.

3.4 – Programming Language
For most of this project, the main language to be used will be Python. Python is an easy-to-use language. The syntax is simple, allowing for rapid prototyping and is easily debugged. Furthermore, the use of Python allows for the use of the vast abundance of available libraries and frameworks. For a project like this one, it would be made simpler with access to libraries like NetworkX and matplotlib which were used during the development. As well as the impressive libraries' python has excellent data handling capabilities allowing for easy data manipulation.

4 – Project Development

In theory, the development of the project can be split into the following several steps:

- The collection of data to be used in plotting the network.
- The manipulation of data so that it can be used.
- The visualization of the network.

However, in practice, this is never the case since it is easier to program the sections that are needed first. For example, when the data is extracted, it can sometimes be easier to implement the visualisation of the data. This allows for manual testing of the program to see what is working and what needs further investigation.

4.1 – Initial Data Collection

It was decided to look for the most important aspect of the application, for this project it was deemed to be the data collection. The data was originally downloaded from aminor.org. Aminer.org is a dataset designed for research purposes with data extracted from DBLP (Computer Science Bibliography), ACM (Association for Computer Machinery), and MAG (Microsoft Academic Graph). Each paper is associated with an abstract, author, year, venue, and title (Aminer, n.d.). Initially, there was no way to access the data contained in the downloaded files. After searching for documentation and tutorials on how to use the dataset A method was discovered. The method was subsequently discovered to require the use of a third-party service called Kaggle.com (MADER, 2019). This was a formatted data set provided by another user, that is available to download. The data was then available in JOSN format. Once access to the dataset had been achieved the next step was to create a program that could automatically sift through the data. An initial examination of the dataset revealed that the dataset includes other attributes including the references. At this point, the focus could move on to the creation of a module to extract the data from the JSON file.

With access to the data available, the next step was to create and test a method to access the data and modify it so that it was usable. A test data file was created called "data.json" so that when the code was tested it did not need to sift through a data set of 1,000,000 lines of data. With the test dataset created a module to format the data was built. A class was designed for easy access to the used data, reducing the time taken to find the correct academic papers and meaning the code only needs to access the dataset each time a new paper is added to the network, it also reduces unnecessary data being stored.

To begin with, the class was a simple wrapper for that dataset and only contained fields for the abstract, authors, title, references, and paper ID. The only method is to print out the instance in a human-readable format allowing to confirm the successful creation of an object. This was in the "paper_class.py".

By using a class, parts of the project would use object-orientated programming (OOP) (Phillips, 2010). People know what objects are and how easy it is to manipulate them since they are tangible "things" that we can touch, taste, sense and feel. In the physical world, humans interact with objects every day like kitchen utensils when eating. In software development, an object is not a physical "thing" to interact with. Instead, we aim to find methods to mimic this as closely as possible. In this sense when using OOP, we create a model of the "thing" requested. In doing this we can assign these model attributes and methods that allow us to manipulate the object as a whole.

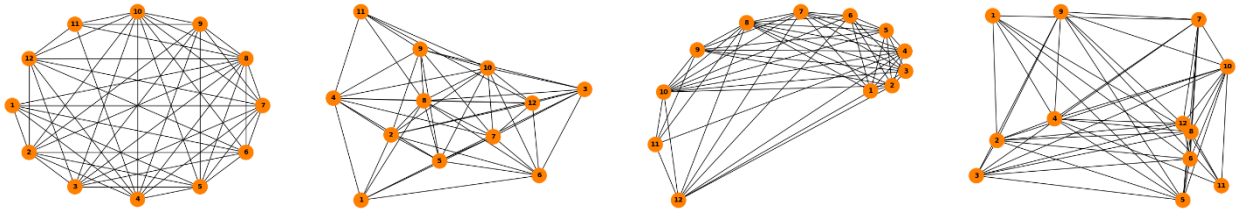
The use of the class, allows us to create methods inside the object that only have a purpose to manipulate the object. One example in this project would be a function to set the gender. This is because all the data is stored in the class so it does not need to access anything outside it. One advantage of this is that when using the built-in method, it is harder to accidentally access the data of another object when manipulating an instance of a class. In this project, an example of this could be updating the author. When using the class, it is harder to update the author field with the name of the author of the paper and accidentally change it to the author of a different paper.

With the paper class being created that left the "middleman" module which was called "JSON_data_paser.py". This was the module that opened the data.json file and extracted the data. It then looped through each line received and used the paper class to make a paper instance which held the data. Doing it with this method made it simple to use and edit if necessary. Since we have used a class here it means that the extraction method

should be kept the same throughout the whole development process. If any changes are needed with the data manipulation it can be handled inside the class methods.

4.2 – Preliminary Visualization Attempt

At this point in the project, the only visualisation attempt was for a proof of concept. Matplotlib was used to visualise the data. As effective as this is for using the data it also prevents the change of the initial paper without restarting the whole code. However, at this point in the development, this was satisfactory. An empty graph was created using the NetworkX libraries graph() function. The list of papers was imported from the data parser module to be iterated through and a node was added for each "paper". At this point, the test data was still being used. Once the program was tested and confirmed that the nodes were being plotted the next step was to add the edges to the graph. Since a Paper class had been created it meant it was easy to access the paper's reference attribute which held a list of the paper Ids that were referenced in the paper. The implementation of the edges was successful. All that was needed to do was test the program. At this point, the program was running therefore a layout method was needed. There were four options for the graph as shown in Fig.1.



A: The Shell layout B: The Spring layout C: The Spiral layout D: The Random layout

Figure 1: Layout options available for the network graph

From this choice, the layout method that was chosen was the shell layout. This was chosen because each node and the edges are visible and traceable allowing for information to easily be implied by the data.

4.3 – Data manipulation - Applying Gender Prediction

At this point, the test data is being used and plotted. But crucially, we need to observe the genders represented in the graphs. To achieve this a new method was developed in the paper class. This set_gender() method took the paper instance as a variable. This allowed access to the information stored inside the class instance. Inside this method, the genders dictionary was initialised. A dictionary was used to store the genders since it is possible for academic papers to contain multiple authors of different genders. Using a dictionary meant the code could keep count of how many male and female authors each paper had. To help with debugging and with the slim chance that the gender predictor cannot work an unsure key was also created. If the gender is unsure, then that means the node will display a different colour. To predict the gender the gender_ai library was imported from GitHub (leophagus, 2017). To use this predictor, it takes the first name only meaning it is necessary to manipulate the string and split it where a space is and take only the first section of the name. With this done, a loop is created to go through every name in the author's array. The gender prediction is then added to the results. With the gender predicted the method to draw the graph is then updated. To do this a colour variable is created as a tuple corresponding to the RGB (Red, Green and Blue) value of the node. Then set the value that represents the gender. To use the ratio, subtract the ratio of male to female from one of the RGB values and add it to another. To calculate the ratio, the code adds the total count of authors and divides one gender by that number. This gives the result seen in Fig.2.

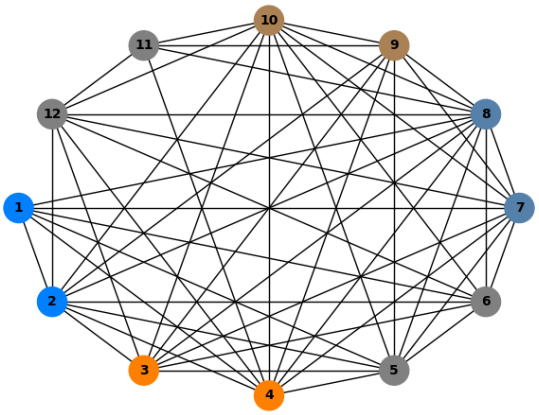


Figure 2: This shows how the nodes can represent the gender with male being the blue nodes and the orange node being female.

Fig.2 shows the graph with the updated code. Nodes 1 and 2 have a single male author, and Nodes 3 and 4 have a single female author. From then a gradient of the colour changes is produced depending on the ratio of male vs female authors. When the ratio is even the colour is grey.

4.4 –Initial Hurdle

The first major hurdle that the project ran into was the attempt to implement the whole dataset rather than the test data into the program. Using this the program would then be able to start searching with actual data. Manual formatting of the data was necessary since when it was downloaded the data was correctly formatted but not readable for Python. The JSON file had to be manually edited, surrounding all the data in square brackets and making each element part of a single list. As all the data was now in a list it means commas needed to be added to the end of every line. However, other problems with the data kept on appearing. These included the data that did not exist (Failed to retrieve the data), the key does not exist (title is not a heading for the data), and the index is out of range. As the problems were solved it became clear that the dataset only included a limited number of references for each paper meaning the dataset was not suitable to meet the aims of this project. Meaning the project was back to square one.

4.5 – Alternative Data Collection Method

It was necessary to find a new method to collect the data, the next idea was to scrape Google Scholar. Before implementing an attempt to scrape Google Scholar, further research was undertaken into Python modules to ease the data collection method. The RESP module from GitHub designed by monk1337 (Pal, 2021) was discovered. This module can fetch all citations of a single paper from Google Scholar and store it in a CSV (Comma Separated Value) file. However, this only retrieves the title and URL. With the URL it is possible to easily access the websites where the scholarly articles are located. This allows scraping the website for additional information. Including the authors, Publication year and more. However, this solution wasn't without issues.

Many websites implement protections from DDoS attacks. A DDoS attack is a distributed denial-of-service attack that is a malicious attack to deny people access to a website by flooding a target server with a request to the website to overwhelm the server infrastructure and crash it.

Protections mean that when making too many requests in a short time it causes the website to deny access to the IP address of the device that is running the requests. This problem can be sidestepped by the implementation of a proxy which reroutes the request through multiple different servers, masking the original IP address and allowing for more requests to Google Scholar. This though may not be enough, so it is a good idea to implement proxy rotation. This is where it automatically rotates through a list of proxy servers after a random number of requests to the server, this helps to avoid being flagged for suspicious activity.

Alternatively, to avoid detection it is possible to implement a rate limit. This helps to avoid making too many requests too quickly. This helps to overwhelm the server and be flagged for suspicious activity. When the program has been flagged for suspicious activity, the server will usually throw the user a captcha to prove that it is a human accessing Google Scholar. However, searching for a method to parse the captcha through to the interface needed to be discovered. Unfortunately, no solutions were discovered.

4.6 – Data Collection Attempt Three

The next attempt to collect data was to search for an API to request from. APIs or Application Programming Interfaces are the fundamental interface between programmers and computers (Arnold, 2005). They are a set of rules and protocols that allow different software applications to communicate and interact with each other (Amazon Web Services, Inc., n.d.). It makes it easier for developers to integrate functionality from one application to another without having to understand the inner workings of each system. It could be thought of as a text message sent from one device to another, with the API being all the protocols of how to transfer the message across.

A short list of possible APIs was drawn up:

- Crossref API
- PubMed API
- Microsoft Academic API
- Semantic Scholar API

After further research on each API, the best one for the aims of this project would be the semantic scholar API. It is simple to use and the data that gets retrieved is easy enough to parse through. Below is an example of how access to the data would be achieved:

https://api.semanticscholar.org/graph/v1/paper/{paper_id}/citations

Using this format, the data can be retrieved from the API and manipulated to build the graph. It was discovered that other developers have also been using this API and one of them had designed a Python module to help with ease of use. This module was called “semanticscholar” and was created by Daniel Silva (Silva, 2023). This is an unofficial Python client that is still being updated. Using this module makes for easier access to the data since it just needs to import the module and access it through the built-in function.

4.7 – Data Collection Implementation

With access to the new method of data collection, unfortunately, means that the data is formatted differently. This means that alterations to the existing code are necessary. The first place that the alterations were made in the class. These took the smallest alterations since the main focus of the class was to standardise the data. The main updates made were to alter variable names, to match that of the data collected. The main alterations produced were in the data parser. This is because accessing the data is different so retrieving and formatting the information is different.

Originally with the new data the first implementation was tested by retrieving one paper. Using the semanticscholar module it was simple to implement and test. Using the Digital Object Identifier (DOI), which is unique for every academic paper. Other forms of IDs can be used when requesting the data which are:

- The PaperId
    - This is the primary identification for the academic papers that are used on the Semantic Scholar website. This is of the datatype string.
- The CorpusID
    - This is the secondary way to identify papers. Semantic Scholars dataset uses these to point to papers. It is of the datatype Int.

With the data requested and received from the API, the next step was to extract and store the data locally to use. When plotting the network graph. To do this the most important query to make was to double-check whether or not the API request was successful. Identifying if there is data to manipulate. To do this an if statement was used to check for the existence of the data. Once it had been confirmed that the data was successfully received the next step was to make an instance of the paper object. Once the object was created it was then time to work on getting the references of the paper. To do this a for loop was implemented to loop through every reference in the academic paper. When looping through each reference, the reference paper is then made into a paper instance and then added to the reference attribute.

4.8 – Visualization Using Actual Data

With the new data collected and formatted correctly, it was then time to attempt to create the network graph. With the new structure and source for the data, it was also time to update the visual interface of the application. Further research on the best modules for graphics with Python. Streamlit (streamlit Inc., n.d.) was discovered. Streamlit is an open-source Python framework that was developed for data scientists and AI/ML engineers to develop dynamic data apps with only a few lines of code (streamlit Inc., n.d.).

With the implementation of Streamlit, it was simple to begin the development of the interface. Within minutes of getting started, the main page was already developed enough to implement a sidebar that can take the academic paper Id as an input and pass it into the data parser and then compile the data it has retrieved into the format that the designer chooses. Building on from the interface the programming to plot out the network graph was implemented.

4.9 – Creating the Graph

The first attempt to create the graph worked by collecting the root paper and then creating it into a paper object. Once it was formatted correctly, it then went on to collect all the referenced papers and appended them to the paper citations attribute. Once they have all been collected and added it would then loop and continue. The first development of this was to do it all sequentially and loop through each reference.

After debugging the code and testing to make sure it all worked, the citations caused suspicions. Every paper in the reference had the same length. After investigating further, it appears that all the papers seemed to be written into every paper citation attribute. After numerous attempts at debugging the cause of the bug could not be found.

This created a segway in design where at the beginning of the function two variables are defined. These variables were two empty lists, one for nodes and one for edges. Using these methods a few alterations were made to the program so that in the correct place the papers got added into the node array. Whilst the code looped through all the references it then also creates a tuple to add to the edges list. The tuple is in the format (source node, destination node).

As the development continued the next implementation was to make the code use recursion so that the same function could be used for all depths. Despite the debugging of the code, it always got stuck into infinite loops. Since a solution was found an imperfect solution was created. A separate function was created called select_depth_to_traverse which would take the depth and paper_Id as an argument. It would then go through an if statement which would call different functions depending on the depth.

After this, there is a set of functions called create_depth_number. These functions were the sequential implementation to create the graph.

4.10 – Network Display

The first attempt to plot the graph continued with the use of matplotlib and altering the current code that had been produced to work with the new data. While this was a successful attempt to plot the graph it had the problem that the created graph was outputted to a different window rather than embedded into the Streamlit application. This is a problem since to change the origin paper the whole program must restart instead of changing the paper Id in the search bar and hitting the generate button. To plot a graph that was embedded in the Streamlit application research was taken into compatible modules. It gave the option of:

- Graphviz – 0.20.3 (Ellson, et al., 2024).
    - An open-source visualization software. Designed to be a module to represent structural information as diagrams of abstract graphs and networks. It is a flexible product and can be used to represent various types of relationships between academic papers. However, the limited number of graph types is restricting the customisation available. If the graph type is not available, then it’s not useful. (Ellson, et al., 2024)

- NetworkX – 3.2.1 (Hagberg, et al., 2023).
    - NetworkX is a Python library that offers a great tool to integrate with other modules. It is easy to use and simple to create graph objects and change how they are displayed. The customization provides multiple benefits which allows for the manipulation of the data to easily show different data patterns that the user is looking for. However, NetworkX was designed for ease of use instead of performance which means when handling large amounts of data performance issues may be encountered. (Hagberg, et al., 2023)
- Plotly – 5.20.0 (Plotly Inc, 2024)
    - Plotly is highly interactive and allows the user to zoom, pan and hover over nodes for additional information. It was designed to be compatible with Python libraries like pandas, NumPy and NetworkX. Despite detailed documentation, there are still gaps in the tutorial and on how to program everything. Therefore when attempting to use it third-party resources may need to be used to establish how to code everything. (Plotly Inc, 2024)
- Pyvis – 0.3.2 (West Health Institute, 2018)
    - Pyvis is a user-friendly and versatile tool for producing a network diagram. It offers extensive customisation options however it has a limited number of layout options, it also has a poorer performance for larger network diagrams. (West Health Institute, 2018)

After completing the research on the different graphing modules, it was decided to use a combination of NetworkX and Plotly. It was decided that NetworkX was to be used to create the graph and be the data structure in the backend and the Plotly would be the front end and be used for the visualisation.

To start with the design, we needed to select the type of Graph that was going to be used so that we could create an instance of the class. The options were:

- Graph
    - This class type implements an undirected graph. It ignores multiple edges between the same two nodes and allows for self-loop edges between the node and itself.
- DiGraph
    - This class type implements a directed graph. It provides all operations that are common to the use of directed graphs. It is a subclass of a graph.
- MultiGraph
    - This is a flexible graph class that allows for multiple undirected edges between pairs of nodes. The additional flexibility leads to some degradation in performance.
- MultiDigraph.
    - The directed version of the multigraph.

For this project, it was decided to use the plain undirected graph. This is because all the academic papers are connected, and it is not important to the aim of the project.

The data structure that the class uses is based on an adjacency list implemented using a dictionary. An adjacency list is a dictionary that contains all the nodes as the keys of the dictionary. The value connected to the keys is a list of all the connected nodes. For example:

G = {‘A’: [‘B’, ’C’], ‘B’: [‘A’, ’D’, ‘E’], ‘C’ : [‘A’, ‘F’], ‘D’ : [‘B’], ‘E’ : [‘B’, ‘F’], ‘F’ : [‘C’, ’E’]}

G here is a graph that contains 6 Nodes called A, B, C, D, E a F. These are the keys of the dictionary. The value of A is a list of B and C. This tells the graph that there is an edge between A and B, A and C. This is the basic structure of how the graph works. However, the actual structure is a dictionary of dictionaries.

The code to display the graph starts with the creation of an empty graph. As the function continues the next part to implement is to add the nodes into the graph. This function is simple since it takes in a list of nodes and edges. To add all the nodes to the graph it was decided to loop through each node parsed into the function and add each node to the graph. In doing this we also added certain important data points into the graph. This includes the title, gender ratio, and the publication date. These data points were decided to be added to the graph because when plotting the graph these are the most common attributes accessed.

The next section of the function was to add the edges to the graph. Looping through the edges list that was parsed into the function. This is done by adding an edge which Is in the format of a tuple of (source node, target node) this then adds the edge to the corresponding node. The next section of the graph that is created is iterating through the pos.items() dictionary where each item represents a node and its corresponding position in the graph layout. In the loop, it then sets the 'pos' attribute of the NetworkX graph to the corresponding position.

The next section of the code uses the plotly module and uses the trace object to represent the edges of the graph. It sets the parameter of:

- X, Y:
    - X and Y are set to empty lists that will be populated by the x and y coordinates of the edges in the graph. Each edge is represented by the coordinates of the source and the target nodes.
- Line:
    - This parameter specifies the appearance of the edge lines. It is set in the form of a dictionary where selected attributes wanted are manually set. In this project we have set the width to 1 and the colour to Gray in the form of hex codes.
- Hoverinfo:
    - This parameter states whether or not to show additional information about the edges if hovered over. In this project, it is set to None so that no information is shown when hovering over it.
- Mode:
    - This shows what the edges are represented by. In this project, it is set to lines.

The next block of code is to add the coordinates to the edge trace. This is accomplished by looping through each edge in G.edges() which is a list of all edges in the NetworkX graph. It then extracts the data from the tuple stored in the nodes which contain the edge position. It sets them into temporary x and y variables and then adds them to the edge trace.

The next step is to add the node trace which contains the following attributes:

- X, Y:
    - Just like with the edge trace these contain an empty list that will be populated with the coordinates for the node in the graph.
- Text:
    - This is an empty array which will contain what will be displayed when hovering over each node.
- Mode:

    - This attribute specifies that the trace (nodes) will be represented by markers in the graph.
- Hoverinfo:
    - This parameter specifies the text that will be displayed when hovering on each node.
- Marker:
    - This parameter specifies the appearance of the markers (nodes) in the graph. It includes several sub-parameters:
        - Size (15) – the size of the node.
        - Showscale (True) – Specifies whether or not to display the colour scale.
        - Colorscale ([[0, 'rgb(0,0,255)'], [1, 'rgb(255,0,0)']]) – specifies the colour scale for the markers. In this project, it is defined to go from blue to red.
        - Color ([]) – An empty list that will be populated with the colours of the nodes based on their gender ratio.
        - cmin (0), cmax (1) – Specifies the minimum and maximum values of the colorscale.
        - Colorbar – specifies the appearance of the colour bar for the markers it includes parameters such as thickness (10), title (gender Ratio), xanchor (left) and titleside (right).
- Line (width = 0) – specifies the outline of a marker

After the node trace has been created the x and y coordinates are populated by looping through the NetworkX graph. For each node, it then needs to extract the x and y coordinates from the graph and insert them into the node trace.

Once all the nodes have been created and added to the trace the colour of the node needs to be set. This is important since the aim is to be able to visualise not only the connections between papers but also to show the gender representation in a particular field. To set the node colour all nodes were looped through the graph and access the nodes attributes, retrieving the gender ratio and publication year. The gender ratio defines the colour of the nodes, and the year defines the transparency of the nodes. The first thing that happens is that it validates that the year exists and if it doesn't it sets it to the minimum year, it then checks that the gender ratio is not none. If the gender ratio is none then the node is set to grey if not, the node is set to the ratio colour.

With the colour of the nodes set the next section creates the title that is shown when hovering over the node, which shows the name of the paper and the number of connections that the paper shares. With the graph produced and all the attributes set it then gets ready to plot the figure.

5 – Testing

Throughout this project, many tests have been run to test the multiple functions that have been built, some of which have to be tested manually and some of which can be automated. Below I will describe the functions that have been automated and explain why I've done this and after I will explain how I've tested it manually.

5.1 – Automatic Tests

Test Get Paper Data

To test the get paper data function a paper Id that would retrieve a positive result was used. The paper Id that was used was 10.1038/s41586-018-0300-2 this is the paper Id for the "Tropical Forest Carbon Cycle and Climate Change" (Mitchard, 2018). Once the paper was retrieved, an assert statement was used to check that the paper data was not none. If it is none then it will return the error message "Function should return data for a valid paper Id". Once it has been checked that the paper exists, it then checks that we can retrieve data from

it by checking that the title matches what was expected. A fake Id was used to test what happens when running the function and force an error. With this, an assert statement was used to check that the returned object is None. If it collects a valid paper, it should then return the error statement “Function should return data for a valid reason”.

Test Get Paper Data Bulk

To test the get paper data bulk a list of paper Ids was created to test the function. The list was parsed into the function. The first thing that was checked was whether or not the papers_data was not “None”. If the data was “None” then an error message would be raised and would state “Function should return data for valid paper Ids” After checking that the data exists, the returned data was checked and should be an instance of a list. If it was not a list, then it would raise an error message stating that it should be a list. With the confirmation that the data was in the form of a list, it then went on to confirm that the length of the list was equal to the number of papers requested. Once the testing for a list of paper Ids had finished, it was necessary to test how the program handles empty lists. With this then we used an assert to test that the returned values were None.

Test Create Paper Object

To test the create paper object function the original test started with a set of mock data. However, when this data was parsed no matter the attempts to debug an error message came up saying that the key used does not exist. After a while of debugging, it was decided to call the get_paper_data function and use live data. With the data retrieved then it was parsed into the create_paper_object function. With the returned paper object, the first test was to check whether or not the paper object is not “None”. If the paper returned was “None” then it would raise an error saying that the functions should return a paper object. Now that it was confirmed that the object exists it was then checked that it was an instance of a Paper class. If it was not a paper object it would raise an error. Once it had confirmed that it was an instance of the paper class it then went on to verify that all the attributes had been correctly formatted.

Test Get Referenced Paper Bulk

To test get referenced paper bulk a paper object was created, it then needed to get the reference paper, so get referenced paper bulk was called and retrieved a list of papers. The first thing checked is that all instances of the list were paper objects. If not all the instances are papers, then an error message would be raised saying that the list should contain paper objects only. Next, it was with a paper with no references. To check that the returned list was empty, we checked that the length of the list was zero. The next this that was tested was what would happen if an invalid ID were entered because this should raise an exception inside the function and should return an empty list. Then to check we just confirmed that the list length was zero.

Test Get Paper Information

To test get paper information it started by testing with a valid paper Id. When calling the function, the returned object was supposed to be an instance of the paper class. If this test failed, then it raised the error message “returned object should be a paper instance”. With the check that test with a valid paper succeeded it then needed to test what happened with an invalid paper Id. This should have returned a “None” item. If it returned anything else, it raised an error message saying that this should be a “None” item for an invalid paper Id.

5.2 – Manual Tests

Some of test are needed to be done manually, this could be because the return no values so there is nothing you can do to check that they confirm the results are correct. Other functions are tested manually since the tests need to check that they are visually correct.

Traverse Network

When testing the traverse network function, it was more difficult to test since it returned no values. When testing it, firstly a couple of papers to do a quick check on whether or not it worked. Next section checked was that the number of nodes in the graph was equal to the number of referenced papers and the original paper. As well as comparing that there was the correct number of edges were created. This should be equal to the number

of referenced papers. After this, the next test that was run was to print out all the titles of the references in the graph and compare this against the references section of Semantic Scholar.

Get Paper Data Cached

When testing this function, it seemed temperamental. The code never seemed to act the same way when run. The application to create the network was run multiple times with the same origin paper. The first run-through takes the full amount of time to create the network with the time dependent on the depth the network was being created. The problems that occurred when not all the papers were being added to the paper cache. Or it was being reset every time it was being called. When testing it was made sure that the function worked when there was cached data, which meant that when recreating a network from the same origin paper it would be recreated more quickly. The next main test was to check whether or not the function still worked when we're collecting data without the paper in the cache.

Create Depth x

The testing of these functions is similar since they are building on each other. For testing with a depth of two, it started with testing using a paper Id that was valid and that the references would work. Once the function had finished. It then printed out the length of the nodes and edges array. Using the length of the nodes array this can be compared to the webpage to find the number of references and add one to it to compare whether or not it is correct. This however is not completely accurate since using the API can only retrieve the academic papers that are in the API. This meant that it was rare that every reference was collected. Using these we then looped through all the nodes to print out the titles of the papers and compared the ones that were missing to the ones on the webpage. It became obvious that the only reason they weren't there was because the paper wasn't stored with Semantic Scholar. This however is not a problem it just means we don't have a complete set,

It became harder to test the other functions since the more and more data collected the harder it was to make sure that it was correct. To help with the testing it became easier to run it through the interface so that it plotted the graph. Using visual recognition, it became easier to tell whether or not it was correct.

The next test to complete was to test it with a paper that was certain to have missing references to see how it handles the missing data. Using the visual interface, it became easy to see that the papers were added, and the code moved on to the next paper.

6 – Description of Final Product

The final product created by the code is an application that takes in the input of a paper Id given to academic papers. The home screen created is a blank page with the title "An app to visualize the representation of gender in academic papers". Depending on the local settings on the user's computer, and web browser settings the sidebar may also be closed. By default, the sidebar should be opened automatically. On the sidebar, this is where the information is input into the application.

The first input field is the paper Id field. It can take one of three Id varieties which are the DOI (Digital Object Identifier), CorpusId or the PaperId. Depending on the Id that the user has entered there can be slight changes in how the results of the application. When using the DOI it is possible to not find the academic paper. This is because not every academic paper is in the Semantic Scholar API. If the DOI is not in the API, then no paper can be retrieved. Since the CorpusId and the PaperId are unique to semantic scholars if they are entered incorrectly then they should always retrieve papers.

The next input is a slider which lets the user control the depth of the graph. In this, the depth of the graph is how many levels of citations that will be collected. E.g. A depth of one is the singular paper. A depth of three would get the origin paper and all its references along with the referenced papers' references. The default of the depth is set to three with a range of one to five.

The next input is in the form of a 'selectbox'. This is a dropdown box which gives three choices for the layout of the graph that is produced. These choices are 'Random layout', 'Spring layout', and 'Shell layout'. This gives the user a choice of what layout they would prefer. The random layout shows the graph with the nodes

plotted randomly with all their edges connecting them. A downside of this is that it is difficult to follow the connections between all the papers. The spring layout shows the graph plotted with the origin paper node in the centre and all the referenced papers branching from them and repeating as they expand. A downside to this is that since there is limited space and a large data set many nodes overlap with each other causing it to be hard to make sure the whole data set is visible. The final layout option is the shell layout which plots all the nodes in a circle with the edges connecting the nodes. This has a similar downside to the spring layout since it is easy for the nodes to be placed on top of each other.

With all the paper id entered and the depth and layout selected the next step is to select the generate graph button. If the paper id is not entered, then a message box is received asking to enter a valid Id. If the id is valid then a box will show up saying generate graph, it will then call on the function to collect all the data. Once the data is collected then the generating graph box will be replaced with Graph generated successfully. If no data is retrieved, then it will say an error has occurred.

Assuming that the data has been collected successfully, the sidebar should then become populated with information about the original paper. This includes the Title, Abstract, Authors, Publication date and URL. With the data collected it's then parsed onto the function that will plot the data. With all the data collected the graph that will be produced will be plotted in the main section of the page ready for analysis. To change the paper, it should be as simple as changing the paper Id and re-pressing the generate graph button.

7 – Evaluation of the Final Product

To best evaluate the product, first one needs to examine the output that the application produces. Below you will find multiple figures produced called Figs.3,4,5 To gather more data and examine the evolution of gender representation three of the graphs will represent the same academic field as well as similar subject topics. After the examination of the graphs, we will then compare the produced results to the aims of the project. Below is the first figure:

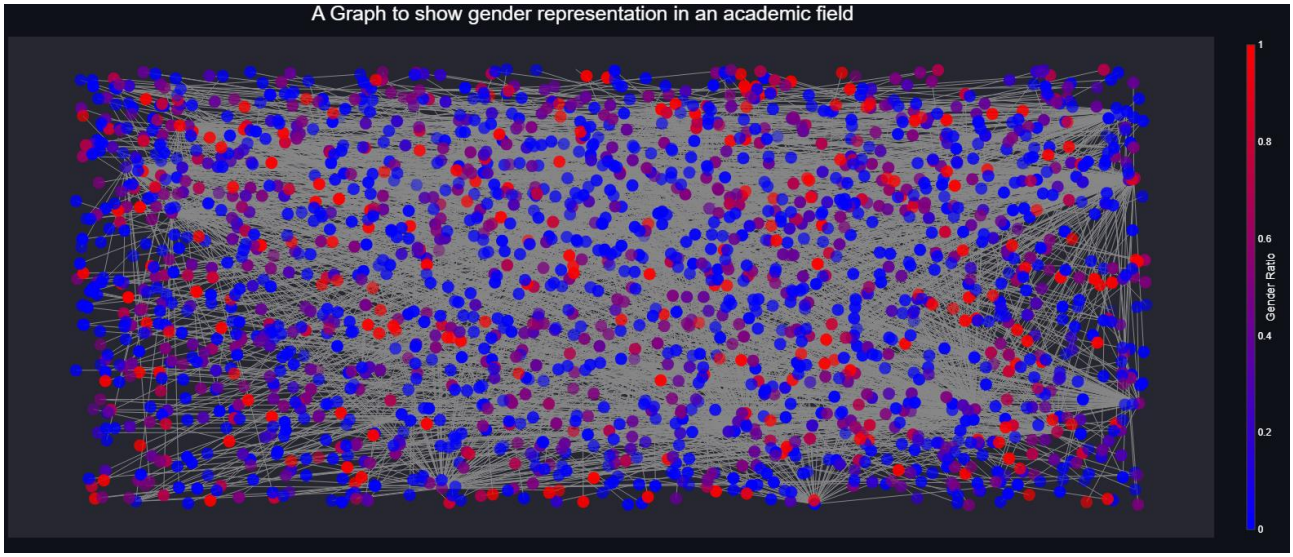


Figure 3: Shows the plot of "Coding as Another Language: A Pedagogical Approach for Teaching Computer Science in Early Childhood" (Bers, 2019).

Fig.3 is a graph produced from the original paper called "Coding as Another Language: A Pedagogical Approach for Teaching Computer Science in Early Childhood" (Bers, 2019). Using this paper, it creates a graph of over 2,000 Nodes (depth of three). At first look the graph appears to be dominant however at a longer glance it seems relatively even. This is because there are more male-dominated nodes along with many nodes that appear to have authors of both genders. The graph seems to contain a ratio of 5 male dominate academic papers comparted to 4 female dominated academic papers.

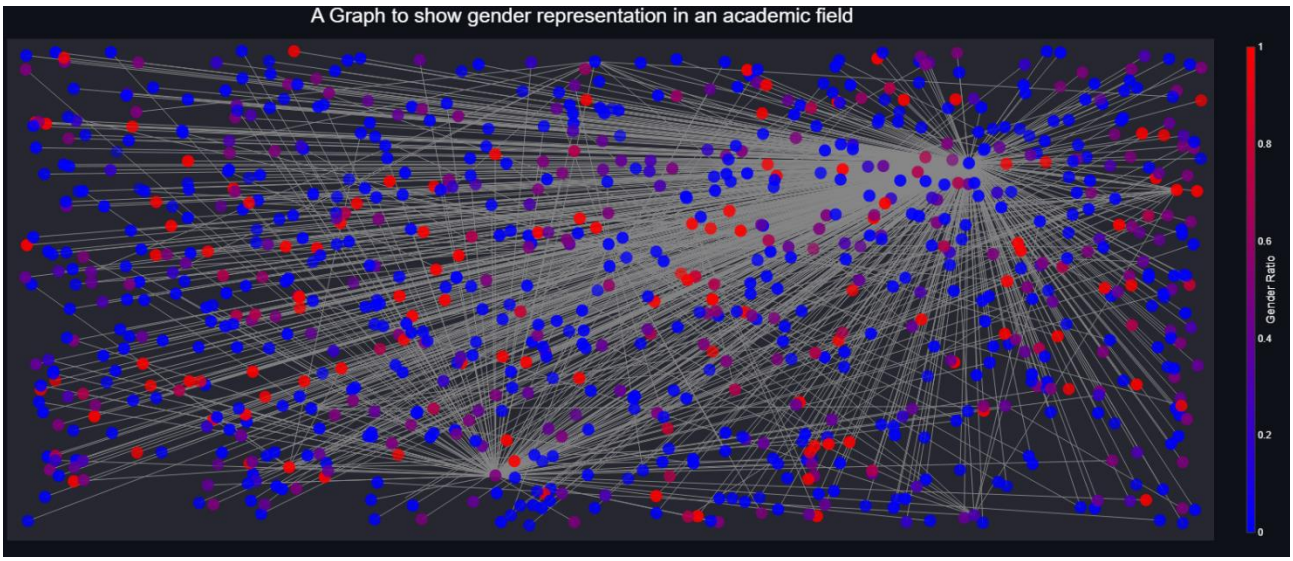


Figure 4: Shows the plot of "An Introduction to Computer Science for Non-majors Using Principles of Computation" (Cortina, 2007)

Fig.4 is a graph produced from the original paper called "An Introduction to Computer Science for Non-majors Using Principles of Computation" (Cortina, 2007). Using this paper, it creates a graph of around 1,500 nodes (depth of three). At a glance, this graph appears to be male-dominated by a higher degree. The graph seems to contain a ratio of 5 male dominated academic papers compared to 3 female dominated academic papers.

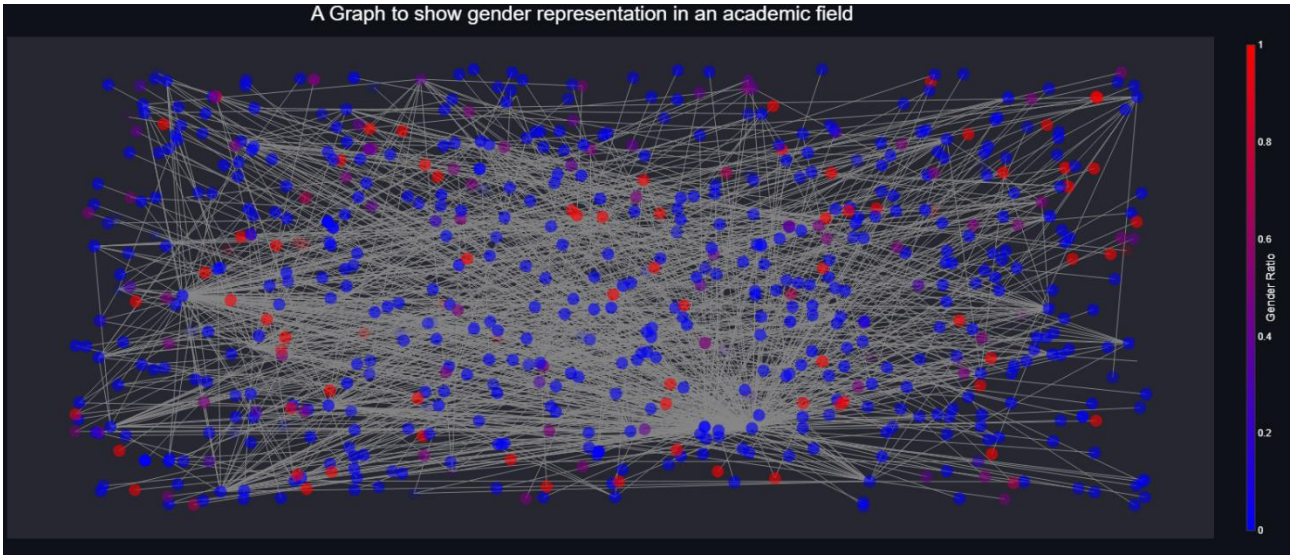


Figure 5: Shows the plot of "Teaching Computer Science with APL: An Introduction to Search Procedures" (Denenberg & Peelle, 1979).

Fig.5 is a graph produced from the original paper called "Teaching Computer Science with APL: An Introduction to Search Procedures" (Denenberg & Peelle, 1979). Using this paper, it creates a graph of around 1,000 nodes (depth of three). At a glance, this graph appears to be male-dominated by a higher degree. The graph seems to contain a ratio of 5 male dominated academic papers compared to 1 female dominated academic papers.

Using the result produces by these graphs an evaluation can be made by comparing the results to the aims of the project. The first aim was to create a network of academic papers, references and citations. This aim could be considered complete. This is because Fig.3,4,5 Shows that a network has been created. The fact that multiple plots has been produced shows that it works for multiple academic papers and does not just work for a single paper, but that the results can be reproduced.

The second aim of the project was to develop a model to reliably ascribe a gender to the authors of the network papers. These graphs suggest that this was successful. This is because if the program cannot assign a gender to a the authors then the node will be coloured grey. The fact that no grey node are visible in any of the figures suggest that the gender prediction was successful.

The third aim of the project was to visually differentiate the produced network of academic papers, not only by gender but also the date of publication (To determine any change in gender representation over time). This aim could be seen as half successful. This is because the plots only differentiate gender of each node. This was not the plan since each node also implements transparency depending on the publication date. This has not worked well since the change in gradient between two years is not enough to be visible.

The fourth aim of the project was to consolidate the output into a graphical method to show any results produced. This aim could be seen to be successful because the graphs are being produces with the gender predicted for the academic papers being correctly interpreted by the graph. However, improvements could be made to make the evolution of gender representation stand out more.

Overall, this makes the project seem like a success since it allow us to show the gender representation of an academic field overall. Despite the fact that the evolution cannot be plotted over a single paper it is easy enough to analyse by looking at multiple graphs. Using the data extracted from Fig.5 roughly had a gender representation of 5 male academics to 1 female academic. It can be seen that the ratio has changed to be increased to 3 females academic over the next 30 years for Fig.4 and continues to increase until it's a ratio of five to four. This shows how it has changed over time.

We can compare this to Fig.6 which is taken from Zippia (McCain, 2022) which show the representation of women in STEM fields, one of which is Computer Science.

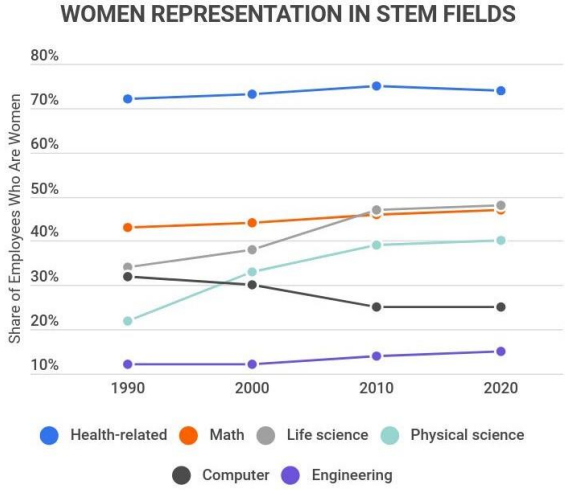


Figure 6: A chart to show the representation of Women in STEM fields (McCain, 2022).

Fig.6 shows that the gender representation this project has produced seems to be higher that that of the data used by this study. It also shows that the trend seems to be going in the other direction. This could be because we are simulating the academic field based on the research papers. This could cause the difference time goes on causing there to be a larger repository of academic papers to reference. Since there are more papers available it means its more likely to select and academic paper that has a female academic. Another reason why this could be different is because of the expansion of the academic field. This could show that the number of women in the academic field is increasing however the number of males in the academic field are increasing at a higher rate.

8 – Critical Assessment of the Project as a Whole

Even though this project worked and could be used to support part of the project goals multiple improvements could be made.

8.1 – Analysis

Despite the successful creation of the network graph, the attempt at creating a method to track the change over time was unsuccessful. This is because it is incredibly difficult to tell the different opacity of each node unless they are plotted next to each other.

A way that this current implementation could be used to calculate the evolution of gender representation is to plot out multiple networks about the same topic but using multiple different papers released at separate times. This can be shown by each graph produced showing different gender ratios, each graph shows a snippet of time. Despite the possibility of the graph from a later date referencing one of the papers used to create the graph from the older date. This would not change the reliability of the results produced since the new graph would only be a small sub-section of the created graph as a whole.

If each of these plots were created from papers released about a decade apart, then one could show the evolution of the academic fields. Because of the method of creating the graph, it can be known for certain that no paper will ever be collected that was released after the original paper.

Even though the evolution can be calculated using this application, it does not necessarily mean that the project was a complete success. This is because all of this needs to be done manually instead of creating it in a way where a list of academic papers is passed to produce the graphs. With multiple graphs produced, they could then be placed next to each other and become easier to analyse.

Another thing that does not help is that the shading makes it harder to analyse the representation. This is because it is difficult to calculate the most common gender that the node represents quickly. Doing this for a small sample size would be simpler but as the graph expands it becomes more difficult to quickly analyse. There could be another method to represent the gender that could be used, for example, plot each node multiple times, one for each author, with each being plotted for the colour of the author's gender. Using this would make it easier to analyse since there is no longer a need to calculate the gender ratio. Another potential method could be to use different-sized nodes.

To improve the overall analysis of the project as a whole more real-world data needs to be collected. This data needs to be collected so that we can compare this project's results to the average of a larger sample size. This will provide a more robust basis for evaluating the reliability of this project and as an assessment of this project's effectiveness as well as help to identify areas where improvements could help. Additionally, the increase in diversity and volume of data will help enhance the validity and reliability of the project, ensuring that the conclusions created are more representative of real-world scenarios.

Overall, this project is effective at creating a visual representation of the network but has a poor performance in making a literal comparison to the real world. To improve on this, it would have been better to use a different graph type. If a bar chart was used it would allow for easier analysis since each year could plot a bar for each gender. This would allow for an easy comparison between the two genders as well as a view of how it changes over time. Another graph type that could be used could be a line graph with a plotting line for the number of authors of each gender and how it changes over time.

However, improvements could be made to the current graph to allow for easier comparison. One simple change could be to keep a legend of the number of nodes for each ratio. Using this it would be simple to read and analyse the data.

8.2 - Improvements

Data Collection

The use of the Semantic Scholar API provides a robust source for data collection however, there are still academic paper that are referenced available that are not stored in the API. This mean we could not be sur how this paper would affect the results.

Alternatively, increasing the depth of the graph may have allowed more data to be included. However, this would increase the amount of time taken to collect the data. This increase in data allow for a more representative average to be calculate meaning a better representation of the whole. However, increasing the depth of the network theoretically also increases the age of the papers. This is because any paper that is referenced will be older than the original paper. This means that for each depth increase the current age for new data points also increases.

Possible solutions to reduce data collection time may include parallelising the code. If the code could collect multiple papers simultaneously, then the process could be sped up and more data could be collected. Another benefit to parallelising the process is that it can use a multi-core processor. This allows each core to handle separate tasks concurrently, maximising the CPU usage and allowing for increased throughput. With the increased throughput it would allow for the graph to go for a further depth and collect more data. With more data comes a larger picture and a more accurate representation of the academic field.

Despite the benefits of parallelising the code, it would have brought some complexities into the playing field. One such thing is race conditions. Race conditions are when two or more threads have to access shared resources. In this implementation of the code, it would be the list of nodes and edges. With this, methods to prevent these from causing further problems need to be implemented.

Another thing that could be optimised is the paper caching. This is since not every time the functions are called with a paper in the cache it is not always found. This could be because the papers are not added to the cache or being retrieved properly. To improve on this, it would be a good idea to store the cache in a persistent storage medium instead of volatile storage. Since the code is currently using a dictionary, it should be a simple solution to add these data to a JSON file. Other types of storage that could be used would be on a database e.g. (MySQL, or Google Firebase). Using a storage type like this would allow for larger graphs to be made since it would stop the need for requesting the same paper multiple times. It would also help when testing the application. This would be an improvement since at the current time the application takes roughly one hour to collect approximately 1000 papers. Using a cache when rerunning the same paper, it could be reproduced in minutes if not seconds.

The final section that could be improved on would be the implementation of the recursion function in creating the graph. There are five separate functions to create the graphs and to increase the maximum size of the graph a new function will be necessary. Using a recursive function would enable any depth as a search criteria. This would require a small change in the interface since the depth slide is currently capped out to a depth of five.

Visual Representation

Another thing that could be improved upon would be how the graph is plotted, despite the graphs showing everything it is not very clear. Since the graph become a tad overpopulated. Despite the overpopulation it was easy to tell the gender ratio, however, it was difficult to tell the opacity of the nodes which represent the year. With the overpopulation, nodes were sitting on top of each other meaning if by change the opaque node is on top of the transparent node it is difficult to see it.

A method to improve the representation would be to give some quantitative data. This could be in the form of a legend containing the number of nodes for each gender / gender ratio. Using this it would allow for a better understanding of the graph. Since this data is absolute, instead of the approximation currently being used.

Another improvement would have been to plot a line graph or scatter graph of the number of male authors and number of female authors on the y-axis plot and a time plot on the x-axis. With two separate lines for each. This may allow the evolution of the gender representation to show more easily however the current graph used shows the whole academic field. Just with a slightly detailed result.

Calculating Gender

Since the project focus was gender, one of the most important aspects is predicting gender. This requires predicting the gender from the author's name. However, this can be challenging when only the author's initial is given, followed by their surname. This makes gender prediction difficult and there is no simple solution to collect the author's name from the academic paper.

One approach to increase the accuracy of the gender would be to search institutional database. This is because every academic paper is released by an institute, if it would be possible to peruse this data base then it could be possible to find the author in the database. This should allow for the access of the full name to use in the gender predictor, unless it contains the academic gender itself.

Evolution

Using the transparency of a node was not an effective method to calculate the evolution over time. This is because as the size of the data set increase it become more likely that there is a paper released in every year between the oldest paper and the newest. This means that the transparency was mapped between these values. If the transparency was the method to be used, then it could have been a good idea to group the papers by decade and map the transparency to the decade released. This would give a greater difference in the transparency making it easier to see the difference.

Aswell as this it means that the graph is visually busy meaning it is hard to recognise when two nodes have differing levels of transapency.

9 – Conclusion

In conclusion, the project aimed to develop an application that visualized the representation of gender in academic papers. This was done by collecting data to create a graph for visualisation and gender prediction techniques. Several key findings were established in addition to several improvements for the future.

Firstly, the project successfully collected data from academic papers by accessing the Semantic Scholar API. However, a small number of paper references could not be collected, since they were not available from the API. This suggests that there is a need for improvement in the data collection method to collect a more complete data set. Data Collection speed could also be improved using methods, such as parallelization and cache optimisation. These changes would allow for more academic papers in the same period allowing for larger graphs to be produced. Despite these challenges, the project demonstrated the potential for deeper insights into gender representation in academia.

Secondly, the graph visualization played a central role in the project, allowing for the analysis of gender ratios among authors across different academic papers and fields of study. While the current graphing method provides a general overview, enhancements such as alternative graph types and addressing problems such as node overpopulation, could improve the clarity and readability of the graph.

Thirdly, the gender prediction revealed opportunities to refine the prediction criteria by incorporating additional author information, like age and location. Despite the challenges of limited data for the author's name, there remains potential for increased prediction accuracy.

Overall, the project represents a step forward in understanding gender representation in academic literature. By addressing the areas discovered for improvement by this assessment, future versions of this application may provide a deeper insight into gender dynamics within academia and provide a model that gives a more accurate representation of the actual field.

Bibliography

Amazon Web Services, Inc., n.d. *What is an API? - Application Programming Interface Explained - AWS*. [Online]
Available at: https://aws.amazon.com/what-is/api/

Aminer, n.d. *Citation Network Dataset: DBLP+Citation, ACM Citation network | AMiner*. [Online]
Available at: https://www.aminer.org/citation

Ana Beatriz Lobo-Moreira, D. G. T. D. S., 2023. Gender representation on environmental sciences editorial boards. *The science of the total environment,* p. 163940.

Arnold, K., 2005. Programmers Are People, too: Programming language and API designers can learn a lot from the field of human-factors design.. *Queue,* V(5), pp. 54-59.

Beauvior, S. d., 1949. *The Second Sex*. s.l.:Alfred A. Knopf.

Bers, M. U., 2019. Coding as another language: a pedagogical approach for teaching computer science in early childhood. *Journal of Computers in Education,* Volume 6, pp. 499 - 528.

Cortina, T. J., 2007. An introduction to computer science for non-majors using principles of computation. *Proceedings of the 38th SIGCSE technical symposium on Computer science education*.

Denenberg, S. A. & Peelle, H. A., 1979. Teaching computer science with APL: An introduction to search procedures. In: *APL Conference*. s.l.:s.n.

Eisenberg, B. & Ruthsdotter, M., 1998. *History of the Women's Rights Movement.* [Online]
Available at: https://nationalwomenshistoryalliance.org/history-of-the-womens-rights-movement/
[Accessed 2024-04-10].

Ellson, J., North, S., Gansner, E. & Hu, Y., 2024. *Graphviz 0.20.3*. [Online]
Available at: https://graphviz.org/

Goldin, C. & Katz, L. F., 2000. THE POWER OF THE PILL:. *NBER Working Paper,* Issue 7527.

Great Learning, 2023. *What is Network Analysis – An overview.* [Online]
Available at: https://www.mygreatlearning.com/blog/what-is-network-analysis/#:~:text=Network%20analysis%20is%20a%20powerful,the%20system%20as%20a%20whole.

Hagberg, A., Schult, D. & Swart, P., 2023. *NetworkX*. [Online]
Available at: networkx.org/documentation/networkx-3.2.1

leophagus, 2017. *gender_ai*. s.l.:Github.

MADER, K. S., 2019. *AMiner Academic Citation Dataset*. [Online]
Available at: https://www.kaggle.com/datasets/kmader/aminer-academic-citation-dataset

Marszalek, J., Kocik, J. & Polok, M. K., 2020. Gender Inequality in Academic Medicine: An Unconscious Bias Perspective. *International Journal of Environmental Research and Public Health*.

McCain, A., 2022. *Zippia "40 TELLING WOMEN IN TECHNOLOGY STATISTICS [2023]: COMPUTER SCIENCE GENDER RATIO"*. [Online]
Available at: https://www.zippia.com/advice/women-in-technology-statistics/

Mitchard, E. T. A., 2018. The tropical forest carbon cycle and climate change. *Nature,* Volume 599, pp. 527 - 534.

Ortiz-Ospina, E., Hasell, J. & Roser, M., 2018. *Economic Inequality by Gender*. [Online]
Available at: https://ourworldindata.org/economic-inequality-by-gender

Pal, A., 2021. *RESP : Research Papers Search*. s.l.:GitHub.

Phillips, D., 2010. *Python 3 Object Oriented Programming*. s.l.:Packt Publishing Ltd.

Plotly Inc, 2024. *Plotly*. [Online]
Available at: https://plotly.com/python/

Silva, D., 2023. *SemanticScholar 0.7.0*. s.l.:s.n.

streamlit Inc., n.d. *Streamlit Docs*. [Online]
Available at: https://docs.streamlit.io/

West Health Institute, 2018. *pyvis - Read the docs*. [Online]
Available at: https://pyvis.readthedocs.io/en/latest/documentation.html