Improving Search Queries for Research Productivity Assessments in Biomedical Research

Honors Thesis

Adam Reynolds

Advisor: Dr. Clément Aubert

In-Field Reader: Dr. Andrew Balas

Abstract:

The findings of this study were built upon research performed by Dr. Clément Aubert, Dr. Andrew Balas, and two former Augusta University students, CJ Tran and Noah Sleeper. Our goal with the current study was to find a solution to a few challenges faced when building a database from existing datasets. The main three challenges faced includes merging data and ensuring all data is unique, dealing with data that overlaps with data from other datasets or existing data in the database, and assessing the impact of research production in a multidimensional space. As research becomes more diverse, it is difficult to determine which metrics are most impactful when analyzing productivity of research in a multidimensional space. We would like to present our solution for these challenges, as well as implementations not yet introduced.

Keywords: Computer Science, Research Productivity, Search Queries, Matching Datasets

Introduction:

As researched and found by CJ Tran, Noah Sleeper, Dr. Aubert, and Dr. Balas, assessing research productively have become increasingly more difficult as databases and their datasets have grown more diverse. Essentially, this means that the quality of research is more difficult to measure with simple attributes. Simple attributes include the number of publications a researcher has made, the number of papers that cited a publication, and grants awarded for the research. We propose a plan to facilitate an approach to better classify research productivity using existing research databases and their datasets to determine the quality of the research. A few attributes are tangible, such as grants received for research done or patents, while other attributes are intangible, such as positive or negative opinions from the general public regarding the research.

The current approach used in the field to measure the quality of researchers involves analyzing a few attributes, such as the authors, involvement of stakeholders, the research methodology, and biases present in the study. In addition to this, the number of co-authors, the quality of co-authors, the number of grants, and the number of citations their research has is also analyzed. Not all research is performed equally, which can be evident in the quality that is perceived using the attributes. However, a few of the attributes used in the previously mentioned approach are extremely difficult to quantify and detect automatically. It is not inherently easy to see whether the stakeholders influenced the study and introduced biases within the research or whether the study was using a biased methodology which influenced the results of the study. Despite being hard to quantify and insert into a database, these qualities are still useful in assessing research productivity.

Databases are structured collections of data, usually stored digitally and accessed through electronic means. Within databases, you can store, manage, and analyze data. The data is typically stored within tables comprised of rows and columns. The columns are typically known as the attributes and have a header that labels them. Examples of attributes include first name, last name, date of birth, and email addresses. Not every database has the same attributes and not every database store data the exact same way. The rows are filled with values that match the columns. A few examples are “John”, “Smith”, 12-04-1996, and fakeemail@gmail.com. The most common language used to manipulate data is SQL (structured query language), but there are a few other ways to store, manage, and analyze data in a non-structured way. For this project, we used SQL for our database.

With the amount of data being produced that fills databases, we know that databases are continuously being expanded with scholarly articles, books, patents, grants, and other academic articles. This presents the issue of attributes being described differently in databases, identifiers not being consistent across databases, and the inability to match data to the correct attributes. Matching datasets has been an issue explored before by Harry Halpin and Fiona McNeill in their publication “*Discovering meaning on the go in large heterogenous data*”. Their article focuses on data being used dynamically and updated dynamically, so their proposal was to discover meaning dynamically as well, hence the name “Discovering meaning on the go”. Since it is hard to derive meaning from data sources alone, it is important to compare data from a variety of sources to better understand and derive meaning from data (Halpin & McNeill, 2013).

Our proposed solution is to gather a set of attributes that are easily quantifiable and tracked within existing databases. This list of attributes includes authors (first and last names), co-authors, their current institution, their past institution, their emails (former and current), as a few other attributes gathered from other databases. Using these attributes and existing datasets, we would create a new database using the multiple facets of information gathered. However, as mentioned earlier, some data overlaps with other databases due to the magnitude of other databases. For example, PubMed includes a plethora of research data that can overlap with data extracted from the National Institute of Health (NIH) databases. Google Scholar, a meta database, combines data under a researcher that seems similar, which may be entirely different research that the researcher has performed. The search queries that a user provides can garner a variety of different information depending on the database, including irrelevant information that the user does not want or information that may be combined into one, as in certain meta databases. Using a variety of these attributes, we are building our own database to include relevant information that qualifies in those categories, as well as establishing a linking system to improve search query results. This linking system would use multiple facets of information within the database to provide a more accurate identification of a researcher or publication that a user is looking for.

Our proposed list of attributes and our proposed database would be able to solve this issue, provided a user provides enough information in their search query. Hence, a new addition to the proposal is a template for what their search query should look like for the best results. As one would expect, the more information that a user provides, the more accurate their search results would be. Researchers using the same name may not have gone to the same institution and there is a high likelihood that they have not shared the same emails. Using all the information in our database, we hope to solve the matching issue between datasets by ensuring that we use all information in our database links a researcher to their data. Their name could link them to their institution, grants awarded, emails, and much more. However, since names are not unique, we decided we needed a unique identifier to ensure that only relevant information would be provided.

ORCIDs are unique identifiers for researchers who have signed up and have their data on the ORCID website. Their goal is that they wish to establish a difference in researchers with the same (or similar) names by providing them an identifier that would allow them to claim their work. However, since researchers must sign up to claim their ORCID, this introduces an issue where researchers who have passed are unable to obtain an ORCID. In addition to this, some researchers do not want to go through the process of signing up or have not had the opportunity to hear about ORCID so they can claim their identity. This unfortunately means that not every researcher will be able to have a unique identifier, for their own reasons. Our solution is to assign an ID that tracks a researcher and does not allow for a researcher to be added into our database twice but does allow for publications and other essential data to be added and linked to that researcher. However, we will include an attribute that links the ORCIDs as well, if they are present for a specific researcher, since this would allow for easier matching of attributes to a researcher in certain scenarios.

Web-scraping plays a vital role in the expansion of our project, and it is important to understand the importance of the search query being used. Some search queries can return irrelevant information that a user does not want to know about, considering they may be trying to get articles that are solely written by a researcher or looking for a researcher based on another facet of information. Since we will be web-scraping from multiple sites, it is important to understand the methods that the sites employ to return information. There are two general methods employed when a search query is initiated on a database. The first method is known as a heuristic search, which is also known as an informed search. The algorithm can take the best path based on the information provided and it is able to provide optimal results for a search query. The second search method is called a blind search, which is also known as an auto search. This method involves using a breadth-first search and searching every possible path for information relevant to the search query. Since it goes down every path, it can be inaccurate and return results which may not fit the exact specifications of a user's search query because it is returning partial matches as well. Luckily, we do not have to specify which type we are using when web-scraping, since most sites try to employ a heuristic search first, and if that fails, switches to an auto search to return results for a user's search query.

Since our program heavily relies on searching for datasets from existing databases online, it is important to understand how our data is being gathered, since it helps derive how we should expect our user queries to look for future use of this program. For the most accurate results, we would provide a template for the users to format their data to get the most accurate results. We would not require them to use it, since it may involve them having to search for necessary information to be able to use our program, but the template would provide them with a way to garner as much accurate information regarding their search query as possible.

Our program makes heavy use of web-scraping to obtain datasets from existing databases, which I was able to experiment within a side-project with Dr. Wendy Burnett. It involves reading an Excel file, extracting data from the Excel sheets, web-scraping for the desired data on PubMed, and writing the information into the desired cells in Excel. Using the aforementioned searches, I was able to narrow down exact publications based on an author's full name and the title of the publication. Through the project, there was a bit more room for improvement in the search queries, when looking for publications from an author. This could include adding the year the publication was published, any co-authors that worked on the publication, and the DOI assigned to the publication.

Our goal is to ultimately create a database that contains data from a variety of sources that can link researchers together to improve search query results, which would garner desired data that would help assess research productivity. Our plan is to implement the aforementioned strategies of web-scraping and linking values by their attributes to improve the search query results, which, in turn, would make it easier to assess the quality of researchers. Our solutions would solve the challenges of ensuring that the data is unique and ensure that users are receiving relevant results when querying information.

Methodology:

GitHub was the primary tool in which we stored our programs and all related files, including an abstract and DOI finder and a program built by Noah Sleeper to web-scrape and gather files from existing databases. An abstract is a summary about what a research project or publication is trying to accomplish or what the study is about, and a DOI is a digital object identifier, which consists of a string of numbers and letters that uniquely identify an article. In addition to this, CJ Tran's publication was accessible to see the research he performed. Using their information ensured a steady learning of how to properly connect SQL, Java, and web-scraping to effectively retrieve data from the web and build a database. The tools used for the programs involved Java, Jsoup, OpenCSV, ApachePOI, GitHub, and SQL.

To briefly explain some of the concepts used in this project, here are a few explanations of the tools used. Java is a high-level programming language that we are using to extract data. Jsoup is a Java library that allows us to acquire a web page as an HTML document, from which we can extract our desired data. GitHub is a hosting service that allows us to create, store, manage, and share our code with others. SQL is a structured query language that allows the relevant data to be sorted into corresponding tables within the corresponding databases, allowing easier access to specified data, as well as a more structured environment for our data. OpenCSV is another Java library that allows us to read data from files with the .csv extension.

To gather data accurately and thoroughly, datasets that contain researcher data need to be downloaded from their respective databases. These datasets can be downloaded in the form of .csv, .xlsx, .pdf, or .xml files. For clarification, .xlsx files are associated with Microsoft Excel files, .csv files are text files with data separated by commas, .pdf files are files that are easily portable and can be opened with a variety of different pdf readers, and .xml files are files used to store data in the form of hierarchical elements using XML tags. Our program would parse the data from these datasets to add them into the database we are creating. The database would need to have distinct data, meaning that once someone exists in the database, the only data that would need to be added is relevant information that does not exist in the database already. To begin the process, our program would need to search through the different types of files mentioned or web-scrape through online databases, which would then create a database based on the relevant information that we want to store. A database is made up of tables that structure the data as columns and rows. The columns represent the attributes, which in our case would include the first name, last name, and other identifying factors for researchers. The rows represent entries in the table, so it would hold the values for the attributes. Databases can have multiple tables, in which they are allowed to share attributes, although it is not necessary. Using this feature, our database would have an attribute related to a specific researcher, allowing us to feature that attribute in our tables.

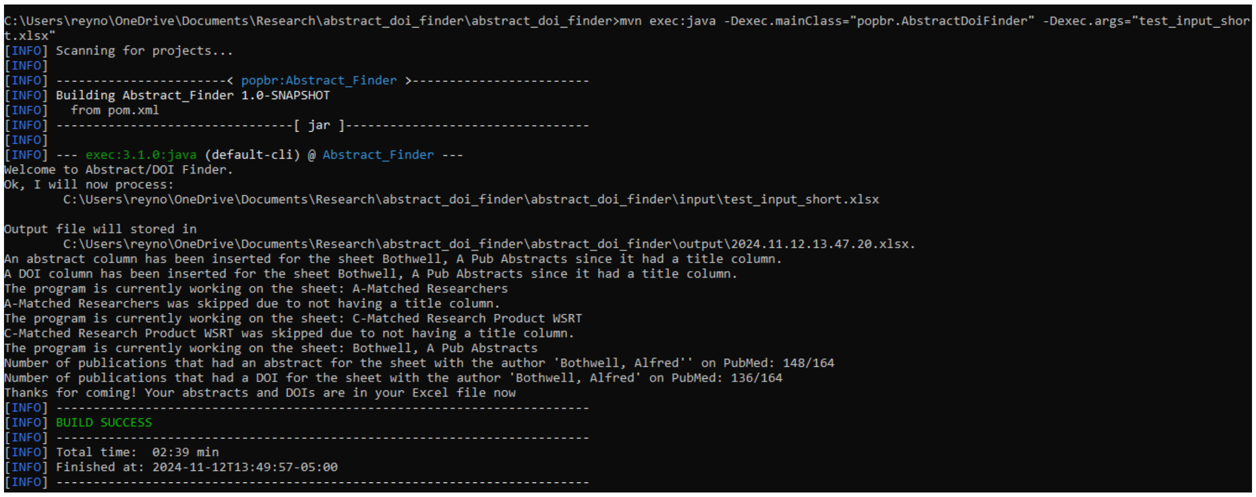
I was not able to start on the project right away, as I did not have the necessary knowledge and skills. Fortunately, I was given an opportunity to work with Dr. Wendy Burnett, as she needed a program to search PubMed and gather the abstracts for hundreds of publications and insert them into a specified column in an Excel spreadsheet. This was an excellent opportunity that allowed me to acquire the knowledge and skills for the project that Noah and CJ worked on. For this, I worked with Dr. Aubert on a program that fulfilled her needs for a program that automated the process. Through the process, we were given the spreadsheet that had all the relevant data. To start, we needed to decide what our search query would entail, so that we could ensure we found the article described. We ultimately decided to use the author's name and the title of the publication since that always queried the correct result for our dataset. However, we recognized that it could be improved if the need ever arose, since we had access to a plethora of data regarding the publications. Throughout the process, we were meeting with Dr. Burnett and we brought up the idea of adding DOIs to her sheets, to which she agreed that it would be a great idea. This introduced a new problem, since we would now need to shift the existing columns to add a column for the DOIs.

We were able to use ApachePOI to parse the Excel file that Dr. Burnett provided us with, extract information from the sheets, create a copy of the workbook, add the desired data to the workbook, and merge the workbooks together to provide the desired results. Jsoup was used to web-scrape information by parsing PubMed web pages as HTML documents that allowed us to search and extract our desired information. Dr. Burnett provided an Excel file that contained titles of publications, names of authors, publication dates, and much more. This helped form the template for other users to use, if they were in need of the program. Ultimately, the template would consist of an excel file that has an excel sheet containing at least an author column with data relevant in the proceeding cells and a publication title column with data relevant in the proceeding cells. A simulation of the program would involve:

1. Accepting an Excel file that followed the specified format mentioned above

2. Parsing through that file to extract an author for the sheet and a list of titles
3. Web-scrape PubMed for the abstracts and DOIs of each publication.
4. Insert the data back into the Excel file. If the abstract or DOI was not on PubMed, the phrase “No abstract/DOI on Pubmed” would be inserted for the respective column.

Figure 1 (below) shows the expected output of the program. It notifies the user of the sheet the program is working on, as well as providing feedback on how many abstracts and DOIs were found per sheet:

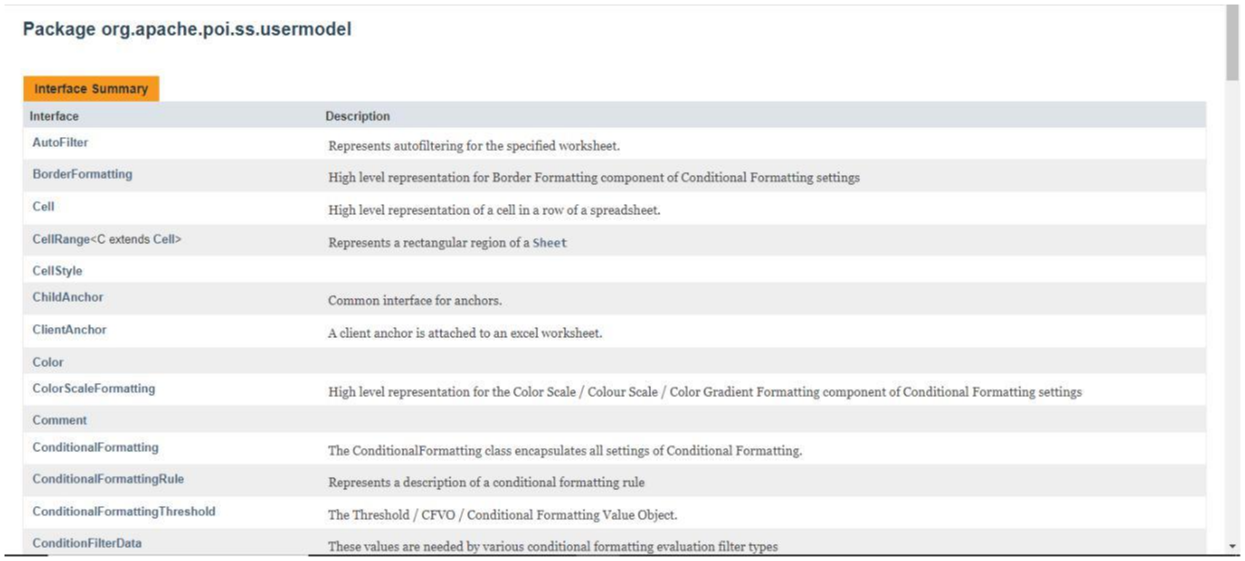


```
C:\Users\reyno\OneDrive\Documents\Research\abstract_doi_finder\abstract_doi_finder>mvn exec:java -Dexec.mainClass="popbr.AbstractDoiFinder" -Dexec.args="test_input_shor
t.xlsx"
[INFO] Scanning for projects...
[INFO]
[INFO] -----------------------< popbr:Abstract_Finder >------------------------
[INFO] Building Abstract_Finder 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec:3.1.0:java (default-cli) @ Abstract_Finder ---
Welcome to Abstract/DOI Finder.
Ok, I will now process:
        C:\Users\reyno\OneDrive\Documents\Research\abstract_doi_finder\abstract_doi_finder\input\test_input_short.xlsx

Output file will stored in
        C:\Users\reyno\OneDrive\Documents\Research\abstract_doi_finder\abstract_doi_finder\output\2024.11.12.13.47.20.xlsx.
An abstract column has been inserted for the sheet Bothwell, A Pub Abstracts since it had a title column.
A DOI column has been inserted for the sheet Bothwell, A Pub Abstracts since it had a title column.
The program is currently working on the sheet: A-Matched Researchers
A-Matched Researchers was skipped due to not having a title column.
The program is currently working on the sheet: C-Matched Research Product WSRT
C-Matched Research Product WSRT was skipped due to not having a title column.
The program is currently working on the sheet: Bothwell, A Pub Abstracts
Number of publications that had an abstract for the sheet with the author 'Bothwell, Alfred'' on PubMed: 148/164
Number of publications that had a DOI for the sheet with the author 'Bothwell, Alfred' on PubMed: 136/164
Thanks for coming! Your abstracts and DOIs are in your Excel file now
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  02:39 min
[INFO] Finished at: 2024-11-12T13:49:57-05:00
[INFO] ------------------------------------------------------------------------
```

Figure 1: Output from the DOI and abstract finder program

Furthermore, Figure 2 showcases a few of the library methods used from the Apache POI library for the program.



Package org.apache.poi.ss.usermodel

Interface Summary

Interface	Description
AutoFilter	Represents autofiltering for the specified worksheet.
BorderFormatting	High level representation for Border Formatting component of Conditional Formatting settings
Cell	High level representation of a cell in a row of a spreadsheet.
CellRange<C extends Cell>	Represents a rectangular region of a Sheet
CellStyle	
ChildAnchor	Common interface for anchors.
ClientAnchor	A client anchor is attached to an excel worksheet.
Color	
ColorScaleFormatting	High level representation for the Color Scale / Colour Scale / Color Gradient Formatting component of Conditional Formatting settings
Comment	
ConditionalFormatting	The ConditionalFormatting class encapsulates all settings of Conditional Formatting.
ConditionalFormattingRule	Represents a description of a conditional formatting rule
ConditionalFormattingThreshold	The Threshold / CFVO / Conditional Formatting Value Object.
ConditionFilterData	These values are needed by various conditional formatting evaluation filter types

Figure 2: Apache POI Documentation

Throughout the months that were dedicated to the abstract and DOI finder program, there were multiple features in the program logic that allowed the program to run smoothly. We added logic to not override user data in the case they already had a DOI or abstract in the cell, copied the user's file before writing to it, and making the program modular to work in cases where the order is not the same as our template. In addition to this, we worked on making the GitHub repository accessible and usable by Dr. Burnett and others who wanted to use the program.

This project was necessary to understand the ApachePOI framework, learn the essentials of Java, experiment with a web-scraping tool, and connect the tools together. Overall, the work of the project with Dr. Burnett allowed for an easier understanding of the program Noah Sleeper worked on and allowed for ideas to bloom about potential search queries our user may provide. Upon completion of this project, I started working on understanding SQL and connecting to a database using Java.

These lead into the next step in improving Noah's program by building databases that incorporate attributes and logic that link the attributes to researchers and their work. Noah's program included a way to create a database, a way to web-scrape from a specified website, a list of attributes to search for in downloaded datasets, a way to parse through different file types and a way to link the tables of the database together. Unfortunately, I was unable to completely build off it and had to rewrite a few parts from scratch to further my understanding of the concepts and use different libraries for better ease of use. For example, I opted to use OpenCSV to read csv files, since it offered a better extraction method than Noah's approach.

To start a little differently, the new task was to design a database that included multiple tables to link researchers, publications, and grants. Figure 3 represents the current database that we have crafted. It is a simple model to demonstrate our idea that multiple attributes could be used to link researchers to their publications, their grants, and how those can be linked to one another. Currently, the researcher table has a researcher identifier, the researcher's name (which contains the full name for right now), an ORCID (if available), and their current institution. The publication table has the title name and a publication identifier. The authorship table has a researcher identifier and a publication identifier, which links an author to a publication, which is helpful in cases where there are multiple authors that worked on a single publication. The grant table includes a grant identifier, a grant name, and the amount awarded from the grant. Finally, the linkage table holds the grant identifier and the researcher identifier, which allows us to link multiple researchers to a single grant, in the case multiple researchers have won the same grant.

For now, the relational model only demonstrates five tables, but we plan to expand this to include tables for institutions and emails, as well as tables that will allow us to link multiple researchers to a single institution and multiple emails to a single researcher. This would allow us to expand the functionality of the program and provide an improvement to the search queries in our database.

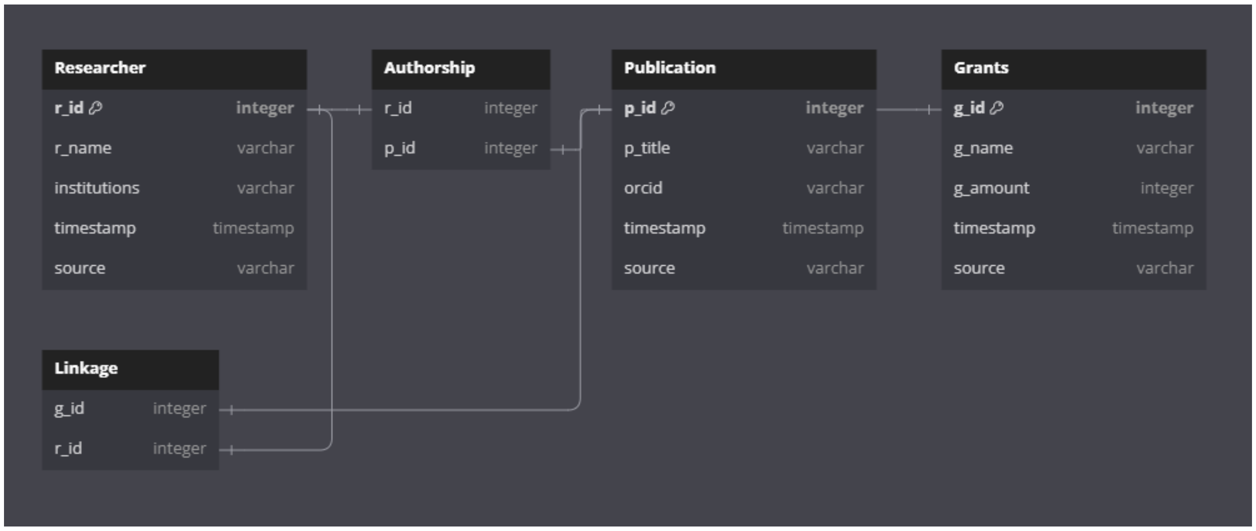
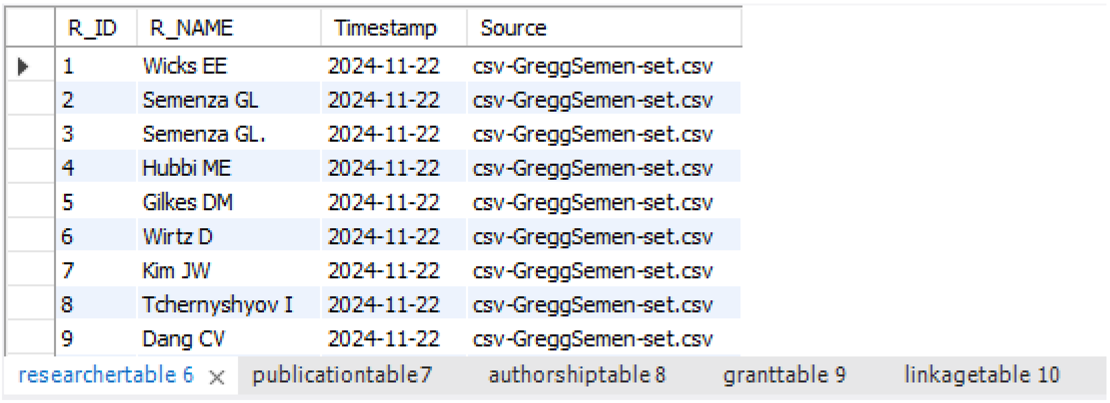


Figure 3: Simple Relational Model for Our Database

The Researcher table tracks a researcher and gives them an ID that will be unique to them, while including their name and institution. The Publication table allows publications to be entered with a unique ID that will allow a title to be added and an ORCID associated with a publication. The ORCID attribute may be added to the Researcher table as well, since an ORCID will be specific to that researcher, if they have one. The Grant table tracks different grants, the amount the grant awards, and a researcher that has won it. The Authorship table exists so that we can link researchers to publications where they may be listed as co-authors. The table uses the unique IDs from the Publication table and the Researcher table to keep track of researchers who have worked on the same publications. The Linkage table serves a similar purpose, as grants can be won by multiple researchers, so it uses the unique ID from the Researcher table to track the researchers who have been awarded a grant. For every table that does not link data together (i.e. the Linkage and Authorship tables), there is a timestamp and source column that will provide the date the data was inserted into our database and the source used for our information. This will be helpful in tracking where information comes from, which will help in assessing research productivity.

Our model expands upon the program that Noah worked on during his time by incorporating the database portion. From his previous work, I was able to acquire ideas for attributes to include in our database, as well as acquire ideas from existing datasets. From there, I have been working on adding data from PubMed and NIH Reporter by extracting CSVs from their websites and extracting data that can be quantified by one of the many attributes in our database. Using the OpenCSV library, I am able to extract the title and the authors of that title from the .csv file extracted from PubMed. Since each .csv file extracted from PubMed will be in an identical format, I was able to find the "column" the titles and authors would be at, keep track of the index the "columns" were at, and extract the data from the rest of the rows.

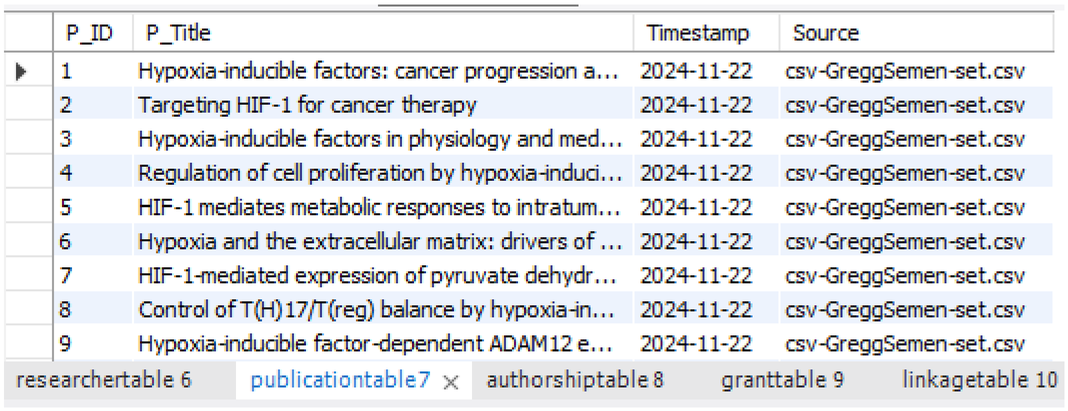
After extracting the data from the .csv file, I was able to connect to the database to get ready to insert the newly extracted data. First, we tested to see if the data we extracted was already in the database. If it was already in the database, we didn't do anything, but if it wasn't, then we added it to the database. Specifically for the authors and titles, we would then retrieve the IDs for the publication and the researcher we just inserted, then link them in the authorship table. By linking them in our database, we now know if multiple researchers worked on publications. Below (Figure 4, Figure 5, and Figure 6) shows how data will be inserted into the table. As mentioned, each researcher ID is unique and is associated with only one researcher, each publication ID is unique and only associated with one publication, and those combine into the authorship table, which link authors to their publications.



R_ID	R_NAME	Timestamp	Source
1	Wicks EE	2024-11-22	csv-GreggSemen-set.csv
2	Semenza GL	2024-11-22	csv-GreggSemen-set.csv
3	Semenza GL.	2024-11-22	csv-GreggSemen-set.csv
4	Hubbi ME	2024-11-22	csv-GreggSemen-set.csv
5	Gilkes DM	2024-11-22	csv-GreggSemen-set.csv
6	Wirtz D	2024-11-22	csv-GreggSemen-set.csv
7	Kim JW	2024-11-22	csv-GreggSemen-set.csv
8	Tchernyshyov I	2024-11-22	csv-GreggSemen-set.csv
9	Dang CV	2024-11-22	csv-GreggSemen-set.csv

researchertable 6 × publicationtable7 authorshiptable 8 granttable 9 linkagetable 10

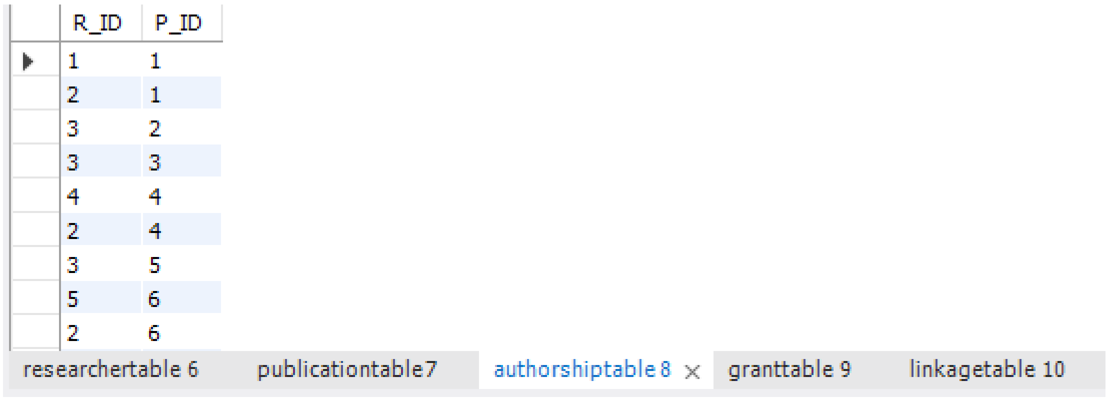
Figure 4: Data Entered into the Researcher Table



P_ID	P_Title	Timestamp	Source
1	Hypoxia-inducible factors: cancer progression a...	2024-11-22	csv-GreggSemen-set.csv
2	Targeting HIF-1 for cancer therapy	2024-11-22	csv-GreggSemen-set.csv
3	Hypoxia-inducible factors in physiology and med...	2024-11-22	csv-GreggSemen-set.csv
4	Regulation of cell proliferation by hypoxia-induci...	2024-11-22	csv-GreggSemen-set.csv
5	HIF-1 mediates metabolic responses to intratum...	2024-11-22	csv-GreggSemen-set.csv
6	Hypoxia and the extracellular matrix: drivers of ...	2024-11-22	csv-GreggSemen-set.csv
7	HIF-1-mediated expression of pyruvate dehydr...	2024-11-22	csv-GreggSemen-set.csv
8	Control of T(H)17/T(reg) balance by hypoxia-in...	2024-11-22	csv-GreggSemen-set.csv
9	Hypoxia-inducible factor-dependent ADAM12 e...	2024-11-22	csv-GreggSemen-set.csv

researchertable 6 publicationtable7 × authorshiptable 8 granttable 9 linkagetable 10

Figure 5: Data Entered into the Publication Table



R_ID	P_ID
1	1
2	1
3	2
3	3
4	4
2	4
3	5
5	6
2	6

researchertable 6 publicationtable7 authorshiptable 8 × granttable 9 linkagetable 10

Figure 6: Researcher and Publication IDs Linked in Authorship Table

Discussion:

The program that I have built has expanded the functionality and coverage of Noah's program, while improving how the tables are linked with one another. Noah's program provides web-scraping functionalities that allow users to scrape datasets, but the exact link needs to be provided. The link provided in Noah's program takes the following format:

"https://www.nsf.gov/awardsearch/download?DownloadFileName="

Following the two types of searches mentioned earlier, we can provide data after the "=" that would allow us to download data depending on the type of data provided. The website provided only allows downloads by year, so the format must follow:

"https://www.nsf.gov/awardsearch/download?DownloadFileName=Year&All=true"

Unfortunately, not every website follows this format, so web-scraping from multiple sources would require the download link to be found and the format to be integrated into our code. Finding the links for all the websites that would be used could prove to be tedious, as it would most likely require manually finding the link for each website we decide to pull from. In the future, we could add a way to web-scrape from multiple sources so that our database stays accurate and up to date.

In addition to this, there are other attributes that are not used in our database currently but can easily be implemented in our tables. This would further improve searching for attributes that are linked together, resulting in the search query results having improved accuracy. Linking the attributes in the way we proposed will allow easier access to information that would improve research productivity by analyzing the relevant attributes proposed.

Currently, the program only takes data from sources provided (whether via web-scraping or manually), so we are unable to provide users a chance to retrieve data from our database. In the future, we want to implement a way that users can utilize the program for their own purposes. This research originally comes from Dr. Balas' questions about improving research productivity with the growing diversity of research databases, so we want the program to be functional at pulling relevant data based on a user's search query. As mentioned before, we plan to have a template available for users to shape their data for the most effective results based on their query. This template would include a variety of different data, which could be linked together for the best results. The way we plan to implement this is via backtracking. Our database hosts data from a variety of sources, so pulling from a table may not retrieve everything the user wants the first time. However, if we find new data which can help identify a researcher in another table, we want to pull from that table again to ensure we pull all relevant data the user may be searching for.

For the matching challenge that we are facing, we currently do not allow duplicates in our database. This allows a researcher to be matched to different attributes, which is possible via the linking strategies mentioned earlier. However, there are a few cases where researchers may not be viewed the same in our database. Our proposed solution is to have a dictionary with verified data in our database, as there may be misspellings or accents in names, titles, and a few other attribute values. Noah proposed using Levenshtein Distance to rectify issues with misspellings in cases where a dictionary would provide incorrect results. Levenshtein Distance is the measurement of the distance between two sequences, in our case, two string values. The output would be how many operations (insertions, deletions, and substitutions) would be needed for two strings to be the same. For example, one dataset would have Gregg Semenza as an entry, which would be inputted into our database. Another dataset may have Greg Semenza, which would be an incorrect spelling of the name, but would appear different in our program. Using Levenshtein Distance, we see that if another “g” is inserted to the end of Greg, it would be the same as Gregg, making the output 1. If the output of Levenshtein Distance is less than a certain metric, we can consider two values the same and link them together in our database, instead of making an entirely new entry for the same person.

Levenshtein Distance could be used in tandem with the dictionary idea, as the dictionary would prove more fruitful in cases where accents are involved in a name. For example, Clement Aubert may be an entry in a data set but may be the same as Clément Aubert. In this case, it would be faster to see if our dictionary had an entry considering these two as the same, instead of using Levenshtein Distance to substitute e and é. In cases where there are misspellings, it would be nearly impossible to consider every misspelling of a name, so using Levenshtein Distance would make more sense.

However, as mentioned earlier, there may be names that are the exact same but reference two different researchers. In this case, our linking strategy would serve the best for this case, as they would have different publications, different grants awarded, and a few more differences which would allow us to sort whether the person we are inserting into a table is already in there or has not been inserted yet.

Finally, we have plans to use GitHub to execute remote actions, which would run our program remotely and update our database accordingly. It would run periodically, which would account for new entries added in databases, so our database would also stay up to date. Running this solution would involve implementing Noah’s idea of pulling datasets from online sources by using the associated web link.

Our proposed solutions for matching would help solve this challenge, as well as the challenges of ensuring all entries in our database are unique. In solving these challenges, it would help improve the accuracy of search results based on a user's query, which would help in cases where users need to analyze researchers based on certain attributes. This would certainly improve research productivity as databases grow more diverse, as well as continuing to be updated.

Conclusion:

In conclusion, our solutions would allow us to solve the challenges faced during this project, although it would require a bit more foundational work to have the model that we're envisioning. Linking researchers together by multiple attributes allows us to be certain that a researcher is who they are in the database and our approach at assigning them a unique identifier in the database will allow us to ensure every researcher in our database is unique. Our proposed linking solution will allow us to solve our matching challenges, as we will be able to use data already gathered in our database to see if a researcher is in our database. There are a few improvements to be made with Levenshtein Distance and dictionary usages in Java, but the foundation of our program will allow us to solve the issue of misspellings, aliases, and accents that appear in other databases. By improving the search queries and implementing the aforementioned solutions, research productivity will become more efficient and have a new metric in which to be measured and analyzed.

References

Aubert, C., Balas, E. A., Townsend, T., Sleeper, N., & Tran, C. J. (2022). Data Integration for the Study of Outstanding Productivity in Biomedical Research. Procedia Computer Science, 211, 196–200. https://doi.org/10.1016/j.procs.2022.10.191

Celeste, R. F., Griswold, A., Straf, M. L., National Goals, C. on A. the V. of R. in A., Education, D. of B. and S. S. and, & Council, N. R. (2014). Measuring Research Impacts and Quality. In www.ncbi.nlm.nih.gov. National Academies Press (US). https://www.ncbi.nlm.nih.gov/books/NBK253890/

Halpin, H., & McNeill, F. (2013). Discovering meaning on the go in large heterogenous data. Artificial Intelligence Review, 40(2), 107–126. https://doi.org/10.1007/s10462-012-9377-4

Levenshtein Distance - an overview | ScienceDirect Topics. (n.d.). Www.sciencedirect.com. https://www.sciencedirect.com/topics/computer-science/levenshtein-distance

ORCID. (2019). ORCID. Orcid.org. https://orcid.org/

Shankar. Blind Search vs. Heuristic Search: Problem Solving - Algorithmic Mind. (2023, December 29). https://algorithmicmind.org/difference-between-blind-search-and1heuristic-search/