OpenAlex Award Data Integration

Lorenzo Bujalil Silva

Forward Data Lab

1 Introduction

1.1 Problem

The problem of data integration stems from the main issue of record linkage in order to establish relationships between new table and existing schema. There first needs to be a definition of how the table will relate to the existing schema and through which data point a join on tables may be done. Record linkage however becomes much more difficult when the data set to be integrated was formulated and maintained by a source different to the source maintaining the larger database. Therefore, a data matching algorithm must be created to be run on the defined data point to find the most relevant record. Data integration is a difficult problem and requires patience and precision to ensure appropriate data integrity.

1.2 Importance

In order to enrich our previously created Academic Database, I was assigned to integrate award data by creating an awards table and formulating relationships to other tables. The awards table should be able to appropriately connect with institutions, authors, and funders. Specifically, the last known institution, primary investigators, and sponsors columns of the award table should allow for connection to the rest of the OpenAlex schema.

Finding connections to the rest of these tables will allow for these institutions, authors, and sponsors to receive credit for the award they received. Integrating this data will also promote better understanding of the academic landscape. Analysis can be done in order to determine why a specific work received an award. If this can be determined then, it may be better understood how to develop better, more relevant research. Funders, Institutions, and Authors may be able to recoginize which fields of study are important to analyze, which can lead to funders and institutions sponsoring more projects in that field of study. Authors may also begin to gravitate to publishing more work in those fields.

The integration of award data is a difficult and complex task, but will further develop the academic database's potential usage and promote academic analysis.

1.3 Difficulty

Data integration is a difficult mainly due to the structure of the data in the award data set and the OpenAlex database. Award data has not been kept under strict standardization which allows for hundreds of different variations of name structure for the primary investigators. There are null values, name placeholders, and complicated naming structures all integrated into the primary investigators column. Preprocessing of this data needs to be done in order to enforce a strict naming standard which allows for easier data matching.

Issues also arise within the OpenAlex database where there are many duplicated records which have been deleted but remain in the database or records exist of authors with their last known institution being a previous institution in their timeline. Also authors in OpenAlex have many different versions of spelling the same author. There is also no naming standard for authors names, which requires to me to enforce the same standardization to the names in OpenAlex to the ones within the awards data set. This issue within the database increases the difficulty of finding records in the data matching algorithm, because many different authors may be matched up to an award record and then a decision needs to be made in order to select the best match.

The most significant difficulty of this project is the number of records that exist within both data sources. There are 2 million award records and around 300 million authors in OpenAlex. Linearly searching for the best data match would make the algorithm to have a quadratic time complexity which will take a very long time given the large quantity of records. Optimizations need to be made in order to reduce searching through all records or to reduce the time complexity of the algorithm.

Finally, a decision needs to be made on how to determine which record is the best match for any given award record. This problem is one of name matching which is one that has been encountered and solved for many years and in many different projects. It is important to decide which one of the methods is the best one in order to be able to create the best matches with the available data, and capable feature engineering.

1.4 Previous Solutions

Data integration is a complex problem that has many moving parts and a variety of subproblems. In the past, work has been in order to make this process simpler and successful. My goal with this project was to take the a combination of works and compile them into my own solution that would work well with this problem.

Ideally, I would have worked to use previously built API's or modules to link the records, but many times these would not be kept up to date and would have required in-depth analysis to be able to fix or understand the undocumented code. Instead, I created my own solution based on the logical steps that other developers/researchers have taken in order to solve this problem.

After doing in-depth research on the problem of data integration, a pattern seemed to emerge from the presented solutions. Each solution presented the following sub problems:

- Data Normalization/Standardization between disparate data sets.
- Data Deduplication to prevent false positives
- Feature Engineering to select most important parts of a name.
- Data Indexing/Filtering to reduce unnecessary similarity calculations.
- Matching Function based on well-known name similarity algorithms.

These five key steps were discussed in many other problems were what I used in order to structure my work in this project.

1.5 My Approach

After determining the general pattern in order to be able to match records together, I decided to formulate my own version of this solution that will suit the needs of my problem. It was very difficult to exactly replicate other developers solutions to this problem, due to the reason that the data was inherently different. There were different edge cases that they needed to handle which were not present in mine, and my data had features which at times required more complicated preprocessing. Overall my solution is complicated, requires many moving parts, and still requires further development to optimize:

- Extract raw award data and store into local database.
- Retrieve data from local database, perform data normalization to handle all cases, build a conformed layer of data to store normalized data.
- Perform feature/data engineering on the conformed data to determine most relevant features to index data.
- Build indexes on display name column data in the Open Alex database to be able to retrieve and manipulate data faster.
- Generate a timeline for each author to be able to further index authors that have their last known institution to be one in the timeline of the given author.
- Build a weighted equation of values calculated through a variety of name similarity algorithms: n-gram, soundex, metaphone, double metaphone, and levenshtein distance. These algorithms were apart of modules installed directly to the PostgreSQL schema.
- Generate a SQL query to send to our Open Alex database to retrieve the most similar data with the appropriate filtering conditions and similarity matching equation.
- Retrieve most similar author and store within a mapped authors data source. This is where data deduplication can then occur, and a table can be created to be able to join the awards table to the rest of the Open Alex schema.

1.6 Key Results

As a result of the detailed approach to solving this data integration problem, I was able to build a data matching algorithm capable of finding the correct match 100% of the time. This calculation was generated by mappings of authors where the found records have a name similarity of above 90% and the institution of the author existed within the timeline of the award data.

These results were also generated on set of 5000 authors in the award data. This is a smaller number because it takes about 2-7 minutes for the timeline to be generated for authors. This is the most significant bottle neck in my algorithm. Once the timeline is generated, an author will statistically always be found. Improving the run time of this algorithm will allow for faster data linkage and a final table can be created to be integrated into the database.

2 Related Work

The main related works that inspired my solution for this project were a variety of different works that focus on data integration. Many different developers and researchers have worked on this problem and have discovered a set of steps to take in order to appropriate record linkage. From the beginning of the semester, I researched novel methods in order to match up names together. The main method that appeared from this matching was one of fuzzy name matching. Fuzzy name matching is the principal used to use different features and structures of a name to detect the most similar name.

The main algorithm that I used in order to be able to detect the most similar name in a database was using tri-gram similarity which was presented in a work created by Peter Gleeson. Gleeson presented a method of using a PostgreSQL extension by the name of pg_trgm. This extension is able easily execute and generate all groups of three consecutive letters in a name. Which can be used then to detect similarity between that of another string. Statistically, proven by Josh Taylor, the n-gram similarity tool can easily match up names together with high percentage of correct matches.

The secondary algorithm that I am using in combination with the tri-gram similarity is the double metaphone algorithm. The double metaphone algorithm was first conceived over a hundred years ago, and was created in order to effectively index the English language. In the case of my problem, I was able to easily use PostgreSQL's extension fuzzystrmatch. This extension has a version of the double metaphone algorithm which I can use to generate metaphones for a name and compare them.

List of works:

- https://www.freecodecamp.org/news/fuzzy-string-matching-with-postgresql/
- https://towardsdatascience.com/python-tutorial-fuzzy-name-matching-algorithms-7a6f43322cc5
- https://towardsdatascience.com/fuzzy-matching-at-scale-84f2bfd0c536
- https://medium.com/bcggamma/an-ensemble-approach-to-large-scale-fuzzy-name-matching-b3e3fa124e3c
- https://towardsdatascience.com/fuzzy-matching-people-names-6e738d6b8fe
- https://medium.com/compass-true-north/fuzzy-name-matching-dd7593754f19
- https://www.crunchydata.com/blog/fuzzy-name-matching-in-postgresql

3 Solution

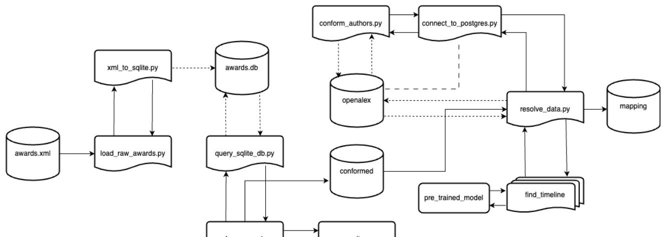

Figure 1: Solution Architecture

4 Results

5 Future Work

6 Conclusion