APPENDIX

1. Background and Data Sources
2. Methods & File Organization
3. Code Components & Functions - Filtering Data/ Formatting Data/ Matching Process
4. Verification Process
5. Results & Conclusion
6. Simple User Guide

**Background and Data Sources**

The initial goal of this project is to identify PTAs, PTOs, and other school-linked non profit organizations and match them to the institutions they have been serving mainly for the years 2016-21. The produced matches would then be utilized for further research and analysis. To extract data and information on school-linked organizations like PTOs and PTAs, we use data compiled in two NCCS databases and GuideStar dataset: the IRS Business Master File (BMF), a monthly updated census of IRS-registered nonprofit associations in the United States based on the IRS Forms 1023 and 1024, and the NCCS Core Financial Files (Core files), an annually updated data source that includes more detailed financial information for larger nonprofit associations that file IRS Form 990, 990-EZ, or 990-PF. Since the NCCS dataset only contains Core files up to 2019, we collected Core files from 2020 to 2021 from GuideStar. Under the Pension Protection Act of 2006, all registered nonprofit organizations must file a tax return each year to maintain tax-exempt status. NCCS updates the BMF data source several times a year to capture new organizations that have recently filed for tax-exempt status and to exclude those whose tax-exempt status has expired. NCCS creates the Core files using IRS Return Transaction Files paired with the IRS BMF. These data sources provide a more complete picture of the financial viability for all tax-exempt organizations, except religiously affiliated ones, raising at least $50,000 annually (as per the new rule established in 2011-12).

To extract data on schools in North Carolina, we use data generated by the Education Data Explorer, an online download tool of the Education Data Portal created by the Urban Institute. We used their data tool to collect data on K-12 institutions at the school level in North Carolina and generated the files based on the necessary variables needed. For each year from 2016-19, we collected the BMF and Core Files along with the schools data for the same year.

To give an overview, we combined the BMF and Core Files for a particular year and performed a filtering process (discussed in detail later) to create a clean dataframe for the organizations we will be matching. We formatted our schools dataset for the particular year in a similar way to keep both datasets cohesive. After both organization and school data frames were formatted to facilitate a more accurate matching, we tested the accuracy of several matching algorithms to perform our current matching algorithm on the two datasets to match organizations to schools.

However, because the NCCS Core Files are only up to 2019, we collected the 2020 - 2021 data for larger nonprofit organizations from GuideStar. Similar to the matching process for 2016 - 2019, we combined the GuideStar data with BMF Files, and then matched to the school dataset. It’s also worth noting that there’s no BMF file for 2021 on NCCS. We used the last file from 2020 and the first file from 2021 instead for the year 2021 as the files are cumulative.

**Methods & File Organization**

The BMF files for years 2016-19 were downloaded directly from the National Center for Charitable Statistics (NCCS) Data Archive: <https://nccs-data.urban.org/data.php?ds=bmf>

Each year had a varying number of BMF files based on the number of times the data was updated in that particular year. For the sake of organization, we downloaded the BMF files as comma separated values (CSV) files and categorized them in separate folders based on the year. The Core Files for years 2016-19 were also downloaded in the same way from NCCS Data Archive: <https://nccs-data.urban.org/data.php?ds=core>. For each year, we downloaded all the files exclusive to that year and excluded the ones that included comparison data to another year. The Core Files were also in a different folder with subfolders that categorized the files by year.

The school data files were downloaded from the Education Data Explorer: <https://educationdata.urban.org/data-explorer/explorer> where we selected the following parameters to generate our files:

* Education level: K-12
* Data level: Schools
* Geography and Timeframe: State (North Carolina) and timeframe > data range > preferred start and end year
* School Characteristics - Click on ‘Select all options’ and under Geographic and identification information, also select all options

Since we can later choose to keep only the variables we want, we can easily download the datasets with all characteristics for the initial stage. Using a similar process, we can generate and download the school data file as a CSVand name the files based on the corresponding year they belong to for clear identification.

After collecting our necessary files, the filtering and matching processes have been programmed entirely using Python programming language and Visual Studio Code as the IDE and main code editor to run files and test the code. Since the original files were substantially large data files, we used Visual Studio Code to easily access and see the data contained in those files but the final product can be run and tested in any code editor convenient to the user. The error rate of our matching algorithms have been computed using a sampling and validation method where we take a random sample of the matches and calculate the accuracy estimate which is the ratio of correct matches in the sample to the total number of matches in the sample (discussed in detail later).

**Code Components & Functions**

This section gives you a detailed rundown of all the different components of our code and what each process entails. Our code makes strong use of Python libraries like numpy and pandas which serve as important tools for our data manipulation and analysis.

To give an overview, we combine the BMF and Core Files containing all educational organizations in North Carolina and then match schools to organizations based on how similar their respective names and addresses are. The code has been divided into four main processes and functions that are utilized in the same order as follows:

1. Filtering Data

The BMF and Core Files are originally extremely dense files and before we use it for intended purpose, it was necessary to extensively filter the files to only keep the variables we need. The *filter\_data* function is the main function responsible for reading in the BMF and Core files, creating the necessary data frames and performing an extensive process to whittle the full array of non-profit organizations down to traditional parent-teacher associations, parent teacher organizations, arts- and sports supporting boosters, and other single-school supporting organizations. The function first reads in all the CSV files located in a particular directory and merges them to a single dataframe. We then filter it by the state of North Carolina. We also filter educational nonprofits using only the National Taxonomy of Exempt Entities (NTEE) classification system codes (in our case, we filtered by NTEE code ‘B’) that still returned any results for school district-wide charities, private foundations, special interest groups, student- and family-serving nonprofits, in addition to organizations that support higher and alternative education. Each organization has a unique employer identification number (EIN) and since our dataframe can contain files from several time points during the same year as mentioned earlier, we remove all other organization entries using their EINs if they have already filed once in that year to avoid duplicates.

After this initial filtering, we used a series of search terms, including PTA, P.T.A., PTO, parent-teacher, school booster, and related variants, to identify non profit organizations that are likely closely connected to schools. Through this extensive process, we categorized each organization in the dataframe by assigning a number code: 1 (PTAs), 2 (PTOs), 3 (Boosters), and 4 (Others). Any organization that didn’t fall into any of these categories was assigned a code of 0 and eventually removed from the dataframe. If future users encounter any other terms that could indicate if an organization is a particular school-linked non profit organization, they can simply add the term to the respective array and the code will assign the corresponding category. This filtering functioning can be easily edited in the future to reflect any changes required by the user.

So, for our first step, we call the *filter\_data* function and input the folder directory (also known as file path) that contains our BMF files for a particular year. The function returns a merged and filtered data frame for organizations included in the BMF files. We repeat this process to create another dataframe for organizations included in the Core Files. For our ‘Name’ variable in the BMF dataframe, several organizations have a more descriptive secondary name. So, for organizations that have a secondary name, we consider that instead of just the name to make the matching process more accurate eventually. So, now that we have our BMF and Core dataframes, we combine them using EINs as a merging factor because several organizations can exist in both the BMF and Core Files. We now have a complete organization data frame called *df\_allorgs* that contains all PTAs, PTOs, and school-linked non profits we want to look at.

1. Formatting Data

After the filtering process, it is important to format our data frames and create variables that will help us access certain data more easily for the final matching process. The *format\_orgs* function is responsible for creating certain variables in the previously created *df\_allorgs* that will be referenced in the matching process. Along with the formatted organization data frame, this function also returns a copy of the *df\_allorgs* data frame so that we do not lose data by constantly changing the original data frame in our matching process. While testing our matching algorithm, we encountered several organizations that had incomplete addresses with just their PO box number which was resulting in some inaccurate matches. To mitigate this issue, the *format\_orgs* function also adds a PO box condition where in organizations that had such incomplete address information were removed from our data frame. Similar to this function, we also have a *format\_schools function* which reads in the school data for that year and converts it into a Python data frame. Both format functions also convert our variables of interest to lowercase letters so that all string formats are cohesive and the matching algorithm produces a more accurate result.

So, for our next step, we input our *df\_allorgs* data frame to the *format\_orgs* function as a parameter which returns a formatted *df\_final\_orgs* along with a copy *df\_final\_orgs\_copy*. Similarly, we input the file path of our schools CSV file as a parameter to the *format\_schools()* function which gives us our final formatted school dataframe *df\_schools.*

1. Matching Process

Our matching process makes strong use of the Python Record Linkage Toolkit which is a library that provides a robust set of tools for linking data records, computing similarity measures and identifying duplicates. The *matching\_process()* function is responsible for all the components of performing the matching algorithm to generate our final set of matches of organizations to schools. Our algorithm consists of two main iterations based on the comparison logic we choose to define a potential match. The first round of matching consists of comparing by name, address and ZIP. We defined the name and address score to be a similarity measure ranging from 0 to 1 where 1 represents that the two variables are exactly the same. So, for example, if a match had a name score of 0.95 then their names had a 95% similarity. The scores are calculated using the library’s in-built similarity measures. The total score for each match is a sum of the name, address and zip scores.

The record linkage method essentially forces a match of every organization to every school. So, for every organization, we keep the match that had the highest total match score which indicates that it was the best possible match for that organization. For the first set of matches, we also restrict the matches to only the ones that had the exact ZIP to make our results more accurate. Our results became more accurate once more restrictions were laid on what constitutes a good match. So, a newly created data frame *potential\_matches1* contains our matches produced in the first round based on name, address, and exact zip.

Moving forward to our second round of matches, we retrieve the indices of the organizations that were already matched in the first round. We now use our *df\_final\_orgs\_copy* dataframe to remove those organizations. This step leaves *df\_final\_orgs\_copy* to only contain those organizations that still need a match which also includes the ones that were removed due to an incomplete PO box address. We perform a similar matching process for this round but the only comparing factor is the organization and school name since these organizations were not matched initially with an exact ZIP or have a complete address. The total match score consists of only the similarity score of the name. We use a variable called match parameter which indicates which comparing factors were used for that match: whether it was both name, address and zip or just name. Now, a new data frame *potential\_matches2* contains our matches produced in the second round based on name.

This brings us to one of our last steps where we input our previously created dataframes *df\_final\_orgs, df\_final\_orgs\_copy and df\_schools* as parameters to our *matching\_process()* function which gives us our first set of matches *potential\_matches1* and second set *potential\_matches2*. We now combine both matches from both rounds by merging the match data frames to create *final\_matches* dataframe. To make our final matches even more accurate, we added one final caveat where we removed the matches that were based on both name and address to only have a name and address similarity score greater than 0.7. Similarly, the matches based on just name to only have a name similarity score greater than 0.7. We extensively tested this by manually checking several samples from the produced matches and identified the incorrect matches. Over 95% of the incorrect matches across all samples had a similarity score less than 0.7 for both name and address. This confirmed our confidence in adding this threshold filter to remove some inaccurate matches.

**Verification Process**

To verify our matching accuracy, we picked Charlotte Mecklenburg county and went through all the schools to see whether they were correctly matched, partially correctly matched, or mismatched for the year 2021 (the most recent data we have). If a direct connection can be identified between the school and organization, such as links on the official school websites, same logo etc., we claim it to be a correct match. If the organization name contains a school name different from the matched school, we claimed it to be mismatched. If the school wasn’t matched to any organization, but after contacting the school, it actually has a PTO/PTA, we categorized it as a mismatch as well.

In 2021, there were 115 primary schools, 30 middle schools, and 33 high schools. 4 of the schools were categorized as missing/not reported and others. When contacting the schools, 1 primary school, 2 middle schools, and 1 high school failed to provide accurate information on whether the school has a PTA/PTO.

Stats:

|  | Elem | Middle | High | Total |
| --- | --- | --- | --- | --- |
| Matched | 87/114=76.3% | 22/28=78.6% | 24/32=75.0% | 133/174=76.4% |
| Partially Matched | 4/114=3.5% | 0/28=0.0% | 1/32=3.1% | 5/174=2.9% |
| Mismatched | 23/114=20.2% | 6/28=21.4% | 7/32=21.9% | 36/174=20.7% |

It is worth pointing out that most of the mismatch is the school wasn’t matched to any organization, but after contacting the school, it actually has a PTO/PTA. However, the PTO/PTAs weren’t in the BMF or Core files in the first place. If we only look at the matched schools, the rate of correct matches is 94 percent. As a result, the relative high mismatch rate is due to the nature of the data rather than our matching algorithm.

|  | 2016 | 2017 | 2018 | 2019 | 2020 | 2021 |
| --- | --- | --- | --- | --- | --- | --- |
| Elem | 827 | 819 | 789 | 801 | 809 | 830 |
| Middle | 251 | 236 | 236 | 250 | 241 | 245 |
| High | 248 | 258 | 256 | 266 | 271 | 320 |
| Orgs | 1762 | 1703 | 1674 | 1772 | 1760 | 1837 |

The way to interpret the table, take 2016 as an example: in 2016, 827 primary schools, 251 middle schools, and 248 high schools were matched to at least 1 organization, and there were 1762 organizations in total that matched to a school.

|  | Elem | Middle | High |
| --- | --- | --- | --- |
| Average Org | 0.607 | 0.629 | 0.854 |

On average, 0.607 organizations were matched to each elementary school, 0.629 organizations were matched to each middle school, and 0.854 organizations were matched to each high school.

**Results & Conclusion**

We produced our final set of matches for each year using our Python codebook. Through a sampling and validation method, we generated five random samples of the matches where each sample size was 10% of the total number of matches. We calculated the accuracy estimate (ratio of correct matches to total number of matches in the sample) for each sample. The average accurate estimate was approximately 92%. After adding our final threshold measure, the accuracy estimate was increased to almost 94%. This indicates that 94% of our generated matches are accurate keeping in mind that there are some organizations that should be matched to schools that probably do not exist at all in our school dataset.

**Simple User Guide**

While our entire code is in Python, it is not necessary for a user to know Python in order to utilize our codebook to generate the required data. The codebook is well-documented with clear steps that indicate how to use it. To give a simple overview, follow the steps below:

* Download and save your required BMF and Core Data Files in a CSV format.
* Download and save your school data file for the respective year in a CSV format.
* Open the code file *pto\_codebook.py* in any Python IDE like Visual Studio Code, Replit, etc.
* The parts that need user input are documented by a *TODO* doc string to indicate what the user needs to do. Complete **steps 1 and 2** in the code and input the right file path (based on the file location in your computer) for the BMF and Core Files respectively.
* Complete **step 5** in the code and input the right file path (based on the file location in your computer) for the school data file.
* After you’ve given the right inputs for the respective data files, the code will read in and format everything as dataframes to further perform the matching process. Finally, complete **step 9** in the code and input a file path based on where you want to save the matches file on your computer.