TaxData

CS 4624
Multimedia, Hypertext, and Information Access

Final Report
December 13, 2022
Blacksburg, VA 24061

Instructor: Dr. Edward A. Fox
Client: Dr. Florian Zach
Author: Maleehah Zafar

Contents

List of Figures

List of Tables

1 Abstract

The Internal Revenue Service (IRS) Form 990 is filed by nonprofit organizations, such as tourism offices, recording financial information, contributors, officers and their positions, grants, and various other information about the organization. Starting in 2011, many of these forms were electronically filed into Extensible Markup Language (XML) format and were made public on the Amazon Web Services (AWS) and IRS websites.

Although this information is available to the public, XML files are difficult to read. There can be over 1,000 lines of tax jargon in the XML files. Dr. Zach realized this and approached Dr. Fox with the TaxData project. The goal of this project is to convert tax data in flat file and XML file format into a readable format and run some preliminary analyses.

TaxData was first introduced into the CS 4624: Multimedia, Hypertext, and Information Access capstone class during the Spring 2022 semester. A group of students worked on this project from January 2022 to May 2022. I have picked up this project for the Fall 2022 semester to make some improvements.

The report outlines my work on the project through: Objectives, Deliverables, Requirements, Design, Implementation, Testing, User Manual, Developer's Manual, and Lessons Learned.

2 Introduction

IRS Form 990 is filed by non-profit, tax-exempt organizations. The various types of Form 990s are [1]:

- Form 990 is filed by a tax-exempt organization that is exempt from income tax, where the total amount the organization received from all sources during its accounting period, known as their annual gross receipt, is over $200,000 [2].
- Form 990-EZ is filed by a tax-exempt organization that is exempt from income tax, where their annual gross receipt is between $50,000 and $200,000. This is a shorter version of Form 990.
- Form 990-PF is filed by private foundations such as a charitable organization established by an individual.
- Form 990-N is filed by a tax-exempt organization that is exempt from income tax, where their annual gross receipt is less than $50,000 [4].
- Form 990-T is used to report any unrelated business income.

Another type of form that is filed in conjunction with Form 990 is a Schedule. A Schedule gives additional information about tax exempt organizations. For example, Schedule H goes into depth about hospital organizations' policies and activities. Schedule I reports any grants or government assistance an organization received. There are many other types of Schedules. The one that was focused on this semester was Schedule J, which reports compensation information of certain officers, like the CEO of a tax-exempt organization [3].

2.1 Objectives

The initial objectives were to use the code from the previous semester and add the Schedule J data to the database that has already been created. Additionally, I was to create a machine learning algorithm to predict which organization is a tourism office.

After a careful look at the database and code created from the Spring 2022 semester, the objectives had to be changed. The new objectives are to create a universal parser that can extract data from Form 990, and its variations, or a Schedule; create and populate a new database with data from Form 990 and Schedule J; and perform some basic statistical analysis.

2.2 Deliverables

The client and team agreed on the following deliverables:

1. Python script to parse either a Form 990, and its variations, or a Schedule
2. SQLite database containing data from Form 990 and Schedule J
3. A Jupyter Notebook with visualizations related to tourism offices
4. Git repository that is well-documented for future groups

2.3 Client

Dr. Florian Zach is an assistant professor in the Pamplin College of Business, specifically the Howard Feiertag Department of Hospitality and Tourism Management, at Virginia Tech. He would like to conduct research on tourism offices in the United States. One major piece he wants to know is what did tourism offices receive from the government during COVID. To conduct this research, it is necessary to look at Form 990, Form 990-EZ, and some of the Schedules, because tourism offices fill out these forms. Having a database that can pull out information from these forms would help Dr. Zach avoid manually looking through the thousands of XML files to find the information he needs.

2.4 Team

Maleehah Zafar is a senior at Virginia Tech majoring in Computational Modeling & Data Analytics (CMDA) and Computer Science (CS).

3 Requirements

Since this project has now been worked on for two semesters, there are a couple of requirements. First is to get access to the Google Drive for this project. There are a number of files in there that go into detail about Form 990, Form 990-EZ, and the Schedules. It also has a couple of READMEs that discuss the goals of this project and the work that has been completed. Next is to create an account, if not already created, on code.vt.edu. This will allow one to get access to the git repository. Finally, one must create an account on the Advanced Research Computing (ARC) website and then get access to the `nonprofit_tax_data` project. This project involves all the raw XML files and database entries from both the Spring 2022 semester and Fall 2022 semester.

4 Design

There was a major design change in this project in the middle of September 2022 when an error was found in the database. The original design was created at the end of August 2022 and the revised design was created during the middle of September 2022.

4.1 Original Design

When the project was in its initial stages, Figure 1 was the original design schema.

The first goal was to revise the database that was already implemented, by fixing faulty code, adding new code to the existing database, and identifying any insignificant files that could be removed.

Next the machine learning algorithm for identifying tourism offices would be created. The steps that would satisfy this goal are researching and selecting a prediction model, creating my own model based on the research, training the model, and then testing the model to assess its quality.

Finally, I was to create a dashboard to show various statistics on tourism offices. This would be done by extracting only data specific to tourism offices in the United States, creating meaningful plots on those tourism offices, and placing them on an interactive dashboard.

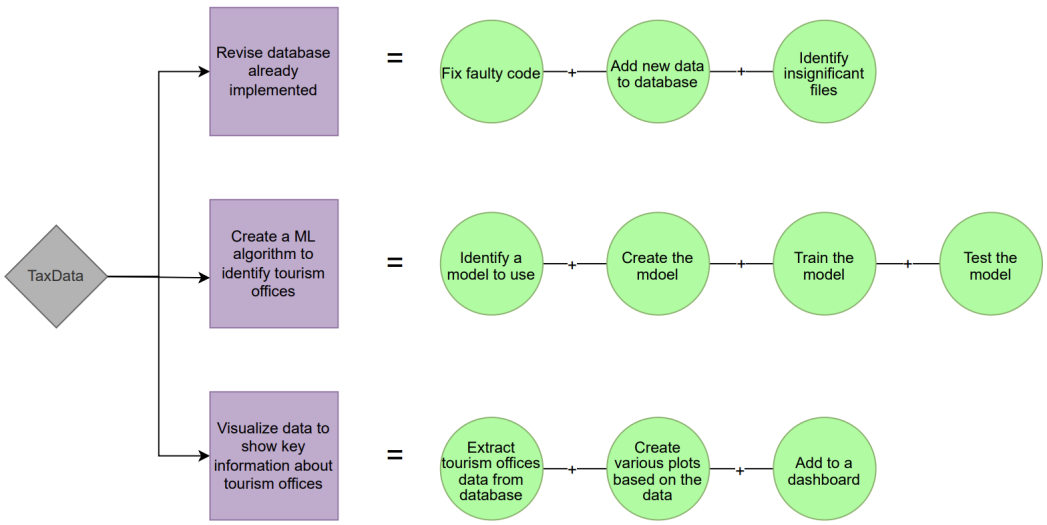

Figure 1: Original System Methodology

4.2 Revised Design

Once the issue was found in the database, Figure 1 was revised and turned into Figure 2.

Creating a universal parser that could extract data from Form 990 (and its variations) and the Schedules is the first step. To accomplish this I need to parse a single Schedule J file and then parse a whole directory of Schedule J files. Once I look at the results to determine if they parsed correctly, I move onto making the code polymorphic so it can parse other types of forms. This can be done by checking if the code works on Form 990. Once the code becomes polymorphic, then I create a new SQLite database and start uploading the data there.

The next goal is to create a Jupyter Notebook with various plots on tourism offices. An example is a plot showing the number of employees and volunteers for each tourism office. The steps for this are the same as the original design.

Once those two goals are accomplished, I need to make sure everything is well-documented by:

- Adding appropriate comments to the Python scripts and Jupyter Notebooks
- Creating a detailed README in the git repository

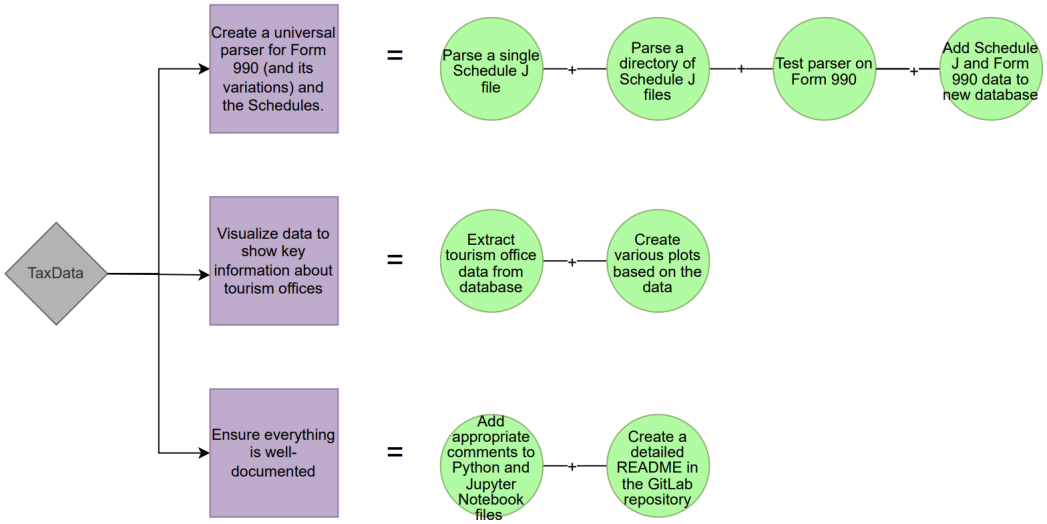

Figure 2: Revised System Methodology

5 Implementation

The universal parser was the main part of the project that had to be implemented. In order to create a parser that could parse Form 990 (and its variations) and the schedules, four steps were taken: creating a database of tags, parsing a single Schedule J file, parsing a whole directory of Schedule J files, and parsing a directory with Form 990.

5.1 Creating a Database of Tags

XML files have two main aspects to them: tags and elements. Tags determine the scope of an element, and appear at the beginning and ending of an element. In Figure 3, books, book, title, author, and genre are all tags. An element is the actual text between the tags. For example, George Orwell is an element in Figure 3.

```
<books>
    <book>
        <title>Animal Farm</title>
        <author>George Orwell</author>
        <genre>Satire</genre>
    </book>
    <book>
        <title>The Invisible Thread</title>
        <author>Laura Schroff</author>
        <author>Alex Tresniowski</author>
        <genre>Biography</genre>
    </book>
</books>
```

Figure 3: Example of XML format

The XML files on the AWS and IRS websites have the same format as Figure 3, but the tags and elements are different. Additionally, a tag can have slight variations depending on the year. For example, Form 990 may have the TaxYear tag in the 2015 form. In 2016, TaxYr would be the tag in Form 990. They both are associated with the same information, but the tag identifier is different.

Initially, I was going to parse through an XML file for every year in order to get all the variations for the tags. However, Dr. Zach got in contact with another researcher who has experience with non-profit data and XML files. Dr. Zach's colleague was able to share his work on all the possible tags that may appear in the XML files and the universal tag associated with it. For example, the universal tag associated with the TaxYear and TaxYr tag is F_00_TAX_YEAR.

With the help of Dr. Zach's colleague, I created a table, TagsMapping, in the new database called NPTR.db, where NPTR stands for nonprofit tax records, and added all the possible tags that appear in the XML files with the associated universal tag.

5.2 Parsing a Single File

Once the table was created, I started to learn how to parse a single Schedule J XML file. Schedule J was chosen first because Dr. Zach is interested in the data in this file, and being able to parse this one would help his research a lot.

There were five steps I took in order to accomplish this:

1. Research Python libraries for XML files. I decided to use xml.etree.ElementTree [5] and lxml [6].
2. Extract all tags from the table that are associated with Schedule J and put this into a list.

3. Loop through the list and check if a tag is in the Schedule J file. If the tag is in the file, then add the tag and its element to a dictionary.

4. Once the list is processed, convert the dictionary into a data frame [7].

5. Look through the data frame to see if the output matches with the file.

5.3 Parsing a Directory

Once I successfully parsed a single Schedule J file, I parsed a whole directory of them. The only major change made was I took the steps from above and created a function that would parse a single Schedule J file. When looping through a directory, I would first check to make sure that Schedule J appears in the file. If it doesn't, then I wouldn't parse that file.

5.4 Parsing a Different File

After parsing a directory of Schedule J files, I moved on to test my code on Form 990. There were some slight differences in the process. For example, when extracting the tags, I would have to look for Form 990 tags not Schedule J tags. This was resolved by creating a function that would find the tags for the form a user wanted to parse. Also for Schedule J, one file could have multiple rows in the data frame because there could be repeated tags. In Figure 3, we see that there are two authors for The Invisible Thread book. Something similar occurs in Schedule J. However, this does not happen in Form 990, so there will only be one row in the data frame for each file.

6 Testing

During my implementation process, it was important that I tested the code to make sure the output was what Dr. Zach desired. There were four different points where I tested my code: when parsing a single Schedule J file, when parsing a directory of Schedule J files, when parsing a Form 990 file, and when parsing another tax form related to Form 990.

6.1 Single Schedule J File

Figure 4 and Figure 5 are examples from a Schedule J XML file. There are other tags in a Schedule J XML file, but I only included the most important ones. Table 1 and Table 2 are what the parser for the single Schedule J file would return. When I was testing whether my parser worked for a single Schedule J file, I had to make sure that it could handle the case when there were multiple officers documented in the file. To test the code for this, I did the following:

1. Find two Schedule J files, one with only one officer (like Figure 4) and another with multiple officers (like Figure 5).
2. Create a spreadsheet with an example of what the parser should return, similar to what Table 1 and Table 2 look like.
3. Run my parser and compare the spreadsheet to my SQLite table. The spreadsheet and SQLite table should have the same output.

```
<EIN>346565597</EIN>
<TaxYr>2018</TaxYr>
<RltdOrgOfficerTrstKeyEmplGrp>
    <PersonNm>MICHAEL L NEAL</PersonNm>
    <TitleTxt>VICE PRESIDENT</TitleTxt>
    <BaseCompensationFilingOrgAmt>0</BaseCompensationFilingOrgAmt>
    <CompensationBasedOnRltdOrgsAmt>402947</CompensationBasedOnRltdOrgsAmt>
    <BonusFilingOrganizationAmount>0</BonusFilingOrganizationAmount>
    <BonusRelatedOrganizationsAmt>73388</BonusRelatedOrganizationsAmt>
    <OtherCompensationFilingOrgAmt>0</OtherCompensationFilingOrgAmt>
    <OtherCompensationRltdOrgsAmt>13597</OtherCompensationRltdOrgsAmt>
    <DeferredCompensationFlngOrgAmt>0</DeferredCompensationFlngOrgAmt>
    <DeferredCompRltdOrgsAmt>200503</DeferredCompRltdOrgsAmt>
    <NontaxableBenefitsFilingOrgAmt>0</NontaxableBenefitsFilingOrgAmt>
    <NontaxableBenefitsRltdOrgsAmt>14406</NontaxableBenefitsRltdOrgsAmt>
    <TotalCompensationFilingOrgAmt>0</TotalCompensationFilingOrgAmt>
    <TotalCompensationRltdOrgsAmt>704841</TotalCompensationRltdOrgsAmt>
    <CompReportPrior990FilingOrgAmt>0</CompReportPrior990FilingOrgAmt>
    <CompReportPrior990RltdOrgsAmt>0</CompReportPrior990RltdOrgsAmt>
</RltdOrgOfficerTrstKeyEmplGrp>
```

Figure 4: Schedule J Example 1

EIN	TaxYr	PersonNm	TitleTxt	...	CompReport
346565597	2018	MICHAEL L NEAL	VICE PRESIDENT	...	0

Table 1: Result of Parser for Schedule J Example 1

EIN	TaxYr	PersonNm	TitleTxt	...	CompReport
346565596	2018	MICHAEL L NEAL	VICE PRESIDENT	...	0
346565596	2018	RICHARD C WENDER	PRESIDENT	...	0

Table 2: Result of Parser for Schedule J Example 2

```
<EIN>346565596</EIN>
<TaxYr>2018</TaxYr>
<RltdOrgOfficerTrstKeyEmplGrp>
    <PersonNm>MICHAEL L NEAL</PersonNm>
    <TitleTxt>VICE PRESIDENT</TitleTxt>
    <BaseCompensationFilingOrgAmt>0</BaseCompensationFilingOrgAmt>
    <CompensationBasedOnRltdOrgsAmt>402947</CompensationBasedOnRltdOrgsAmt>
    <BonusFilingOrganizationAmount>0</BonusFilingOrganizationAmount>
    <BonusRelatedOrganizationsAmt>73388</BonusRelatedOrganizationsAmt>
    <OtherCompensationFilingOrgAmt>0</OtherCompensationFilingOrgAmt>
    <OtherCompensationRltdOrgsAmt>13597</OtherCompensationRltdOrgsAmt>
    <DeferredCompensationFlngOrgAmt>0</DeferredCompensationFlngOrgAmt>
    <DeferredCompRltdOrgsAmt>200503</DeferredCompRltdOrgsAmt>
    <NontaxableBenefitsFilingOrgAmt>0</NontaxableBenefitsFilingOrgAmt>
    <NontaxableBenefitsRltdOrgsAmt>14406</NontaxableBenefitsRltdOrgsAmt>
    <TotalCompensationFilingOrgAmt>0</TotalCompensationFilingOrgAmt>
    <TotalCompensationRltdOrgsAmt>704841</TotalCompensationRltdOrgsAmt>
    <CompReportPrior990FilingOrgAmt>0</CompReportPrior990FilingOrgAmt>
    <CompReportPrior990RltdOrgsAmt>0</CompReportPrior990RltdOrgsAmt>
</RltdOrgOfficerTrstKeyEmplGrp>
<RltdOrgOfficerTrstKeyEmplGrp>
    <PersonNm>RICHARD C WENDER</PersonNm>
    <TitleTxt>PRESIDENT</TitleTxt>
    <BaseCompensationFilingOrgAmt>0</BaseCompensationFilingOrgAmt>
    <CompensationBasedOnRltdOrgsAmt>457111</CompensationBasedOnRltdOrgsAmt>
    <BonusFilingOrganizationAmount>0</BonusFilingOrganizationAmount>
    <BonusRelatedOrganizationsAmt>73082</BonusRelatedOrganizationsAmt>
    <OtherCompensationFilingOrgAmt>0</OtherCompensationFilingOrgAmt>
    <OtherCompensationRltdOrgsAmt>21013</OtherCompensationRltdOrgsAmt>
    <DeferredCompensationFlngOrgAmt>0</DeferredCompensationFlngOrgAmt>
    <DeferredCompRltdOrgsAmt>40992</DeferredCompRltdOrgsAmt>
    <NontaxableBenefitsFilingOrgAmt>0</NontaxableBenefitsFilingOrgAmt>
    <NontaxableBenefitsRltdOrgsAmt>14408</NontaxableBenefitsRltdOrgsAmt>
    <TotalCompensationFilingOrgAmt>0</TotalCompensationFilingOrgAmt>
    <TotalCompensationRltdOrgsAmt>606606</TotalCompensationRltdOrgsAmt>
    <CompReportPrior990FilingOrgAmt>0</CompReportPrior990FilingOrgAmt>
    <CompReportPrior990RltdOrgsAmt>0</CompReportPrior990RltdOrgsAmt>
</RltdOrgOfficerTrstKeyEmplGrp>
```

Figure 5: Schedule J Example 2

6.2 Directory of Schedule J Files

Once I confirmed that my parser worked for a single Schedule J file, I turned this into a function that could parse a whole directory. In order to test for this, I compiled a list of Schedule J files in a directory. The next step I did was extract the Employer Identification Number (EIN) from each of the files. I then ran my parser on the directory of Schedule J files. Finally, I extracted the EINs from the SQLite table and compared them to the EINs I extracted from the Schedule J files to make sure they are equal. Since I know that the parser can parse a single file correctly, I had to make sure that the parser wouldn't skip over any of the files. Checking the EINs to make sure they are in the SQLite database ensures that the parser is extracting data from all the Schedule J files.

One thing I did notice was there were duplicate rows in the database. To combat this issue, I created a Python script that would delete any duplicates. This will be discussed more in section 7 and section 8.

6.3 Form 990 File

When checking if the parser worked for Form 990, I followed the same testing strategy for Schedule J. I first checked if the parser worked on a single Form 990 file. There were some slight changes that had to be made to the parser in order for it to work for Form 990 as well. Once these changes were implemented, I tested the parser on a Schedule J file again to make sure it still works for that. I then parsed a whole directory of Form 990 files with the same testing strategy I used for a whole directory of Schedule J files.

6.4 Other Tax Forms

For Schedule J, I needed to make sure I had multiple rows per file, if necessary. For Form 990, Form 990-EZ, Form 990-PF, Schedule A, Schedule B, etc., there should only be one row per file. Since this is the case, I decided to see if my parser now works for the other tax forms as well because the logic for parsing Form 990 should be the same logic for parsing the other tax forms as well. I tried this parser on the other tax forms using the same testing strategy mentioned above, and my assumption was correct that the logic for Form 990 works for the other tax forms.

7 User Manual

Dr. Zach will have to continue adding to the database once new tax files are released, so I needed to make sure this parser would be run with little to no interaction with the code. There are a couple of things to note about running the parser.

7.1 Running the Parser

Dr. Zach and I have already run the parser on all available tax files for Form 990, Form 990-EZ, and Schedule F. However, this parser still needs to be run on Form 990-PF and the remaining Schedules. There are two different ways to run the parser depending on the tax file.

7.1.1 Parsing Form 990 (and its variations)

If one wants to parse Form 990, Form 990-EZ, or Form 990-PF, then they should run the following command: `python3 ParseTaxData.py year form form_type form_part`. The meaning of these variables are:

- `year` is associated with the directory name in the nonprofit_tax_data project
- `form` is F990 for Form 990, Form 990-EZ, or Form 990-PF
- `form_type` is either PC for Form 990, EZ for Form 990-EZ, or PF for Form 990-PF
- `form_part` is 01, 02, . . . , 11, 12

7.1.2 Parsing the Schedules

If one wants to parse any of the schedules, the following command should be used: `python3 ParseTaxData.py year form`. The meaning of these variables are:

- `year` is associated with the directory name in the nonprofit_tax_data project
- `form` should have the following format: SCHED-A. It should always include SCHED and make sure it is capitalized and there should be a hyphen as well. The A needs to be changed to whatever schedule you want.

7.2 Deleting the Duplicates

Once the parser is run, there are possibilities of duplicates in the tables. The next step is to delete these duplicates by running the following command: `python3 DeleteDuplicates.py`.

For all of these commands, they will be included in the RunPyScript.sh file. This file allows you to run a job in ARC. You will use the following to run this file: `sbatch RunPyScript.sh`. Once this is run, a file that starts with 'slurm' will automatically be created and you can see the progress of the parser in this file.

7.3 Visualizations

Once the parser is run and the duplicates are deleted, you may need to update the visualizations. There is a Jupyter Notebook that does some analysis on the data that was extracted from the XML files.

One looks at the top 10 highest paid officers in tourism offices for each available year. This is shown in Figure 6 and Figure 7. Figure 6 shows the top 10 highest paid officers in tourism offices for all the available tax years. Figure 7 shows the top 10 highest paid officers in tourism offices for the tax year 2018. You can hover over a bar in the plots, as can be seen in Figure 7, and get the EIN for that person and the exact amount they received from the tourism office. One can change the year by clicking on the drop down menu in the top right of the plot.

The second visualization looks at the number of employees and volunteers for each tourism office for every year. In Figure 8, we are looking at the number of employees and volunteers for Hawaii Visitors & Convention

Bureau. Similar to Figure 6 and Figure 7, you can hover over the bar and see some more information like the exact number of employees or volunteers. Also, you can change the tourism office using the drop down menu in the top right corner.

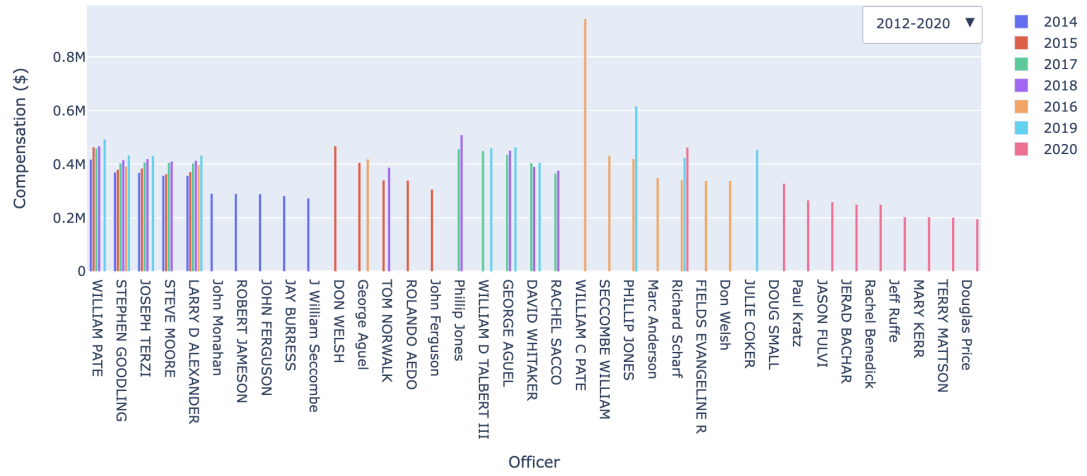

Figure 6: Top 10 Highest Paid Officers from Tourism Offices from 2012-2020

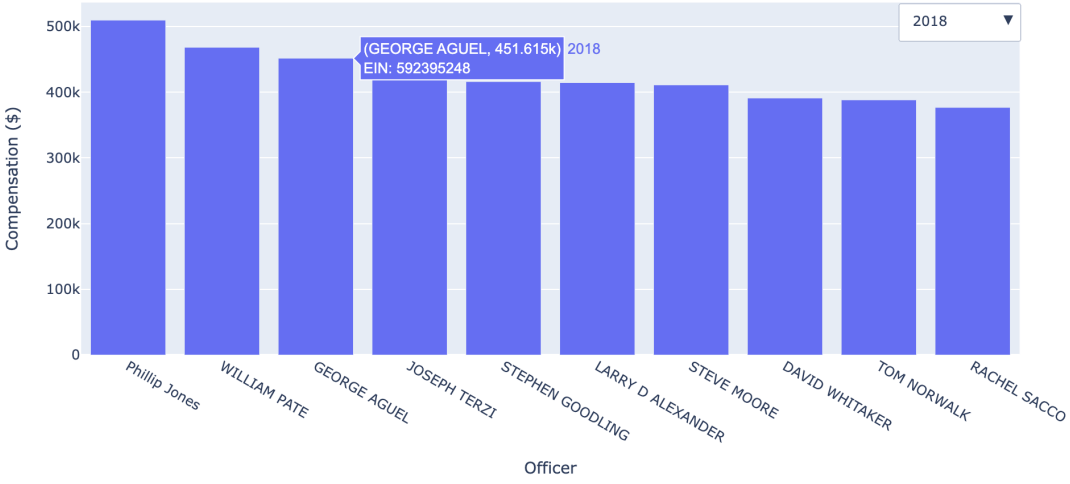

Figure 7: Top 10 Highest Paid Officers from Tourism Offices in 2018

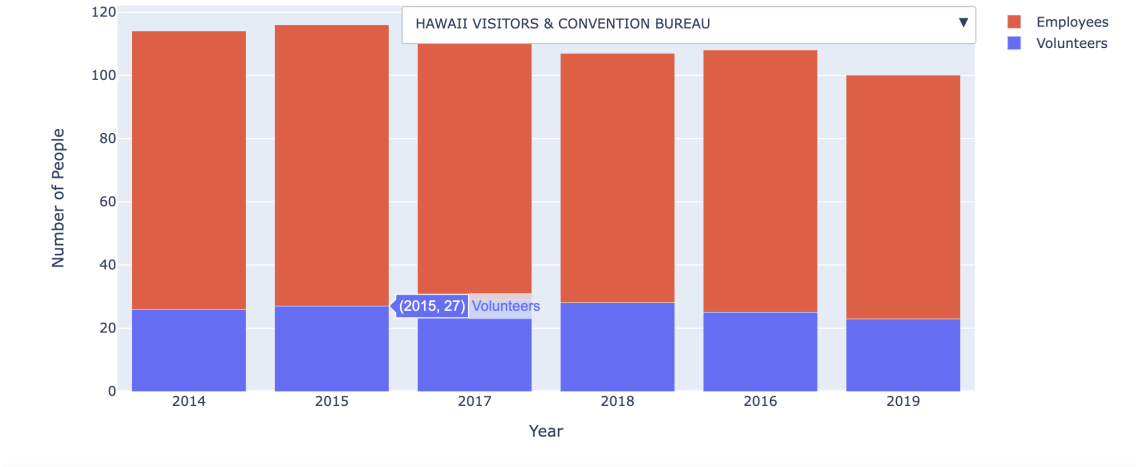

Figure 8: Number of Employees and Volunteers at Hawaii Visitors & Convention Bureau

8 Developer's Manual

In order to continue with this project, there are three places you need to have access to.

8.1 ARC Project

Advanced Research Computing (ARC) at Virginia Tech is a resource for large-scale data storage [8]. Dr. Zach allocated space here in order to store the parsed data from the XML files. The project that Dr. Zach created on ARC is called nonprofit_tax_data and the directory contains what is shown in Table 3.

File/Folder	***Description***
AWS	This is a folder that contains the raw XMLs from the AWS website.
IRS_Website	This is a folder that contains the raw XMLs from the IRS website.
Parser_Log	This is a folder that contains files documenting any errors when parsing the XML files.
990_metadata-master.zip	This zip file was created by the team in Spring 2022, but I am unsure of what exactly is in this because there is no documentation for what this is.
eo_final.zip	This contains eo.db which is the SQLite database that the Spring 2022 team created.
NPTR.db	This is the SQLite database that I created this semester. It currently has data for Form 990, Form 990-EZ, and Schedule J.
unique_EIN.txt	This file contains EINs of tourism offices.

Table 3: Files and Folders in ARC Project

8.2 Google Drive

Dr. Zach created a Google Drive project that contains some important information about Form 990, Form 990-EZ, Form 990-PF, and the Schedules. There are a couple of READMEs in this project that detail what exactly Form 990 is, how to access the raw XMLs from the AWS and IRS websites, how to identify tourism offices, and helpful links in order to learn more about Form 990, its variations, and the Schedules. This project contains CSV files of all the possible tags from the XML files for all the Schedules, Form 990, Form 990-EZ, and Form 990-PF along with the universal tag associated with it. There is also the same unique_EIN.txt file that was in the ARC project. Lastly, there is a spreadsheet that documents how long the parser takes per year, and for each form. This gives a good idea of how much time you would need to allocate for a job if you wanted to run the parser multiple times.

8.3 GitLab Project

There is a repository in GitLab [9] where all the code is located, called nonprofit-tax-data. Table 4 gives details of the purpose of each file or folder in the repository.

8.3.1 TagsMapping.ipynb

If the nonprofit_tax_data project in ARC is empty, which it most likely isn't, then the first step would be to run this Jupyter Notebook. This Jupyter Notebook collects all the possible tags found in the XML files and maps all of them to a common tag. For example, TitleTxt and TitleText tags would be mapped to Title tag.

8.3.2 DeleteDuplicates.py

DeleteDuplicates.py is a Python file that deletes any duplicate rows in any of the tables in this database. The logic for this is:

1. Connect to the NPTR database.
2. Extract all the tables, excluding the TagsMapping table.

File/Folder	*Description*
Spring_2022	This folder contains all the work that the group from Spring 2022 did. Their code isn't well-documented, so it is hard to understand what exactly they are doing. It is a good reference, but I wouldn't recommend running it as what it does is unknown.
TagsMapping.ipynb	A Jupyter Notebook that creates the TagsMapping table in our NPTR database.
DeleteDuplicates.py	A Python script that deletes any duplicate rows from our NPTR database.
ParseTaxData.py	This Python file is the parser.
README.md	This gives detailed instructions about what is in the repository.
RunPyScript.sh	Since we are using ARC, this file is how you can submit a job.
Visualizations.ipynb	This Jupyter Notebook contains all the visualizations that were created and shown in section 7.
unique_EIN.txt	This file contains EINs of tourism offices.

Table 4: Files and Folders in the GitLab Repository

3. Loop through each table in the database and delete any duplicate rows.
4. Save the changes to the database and close the connection to the database.

An important thing to note is that this will keep one row; deleting the duplicates does not mean it will delete all duplicate rows. One row will be kept because we need that row.

8.3.3 ParseTaxData.py

ParseTaxData.py contains the parser. The function parseFile contains all the logic to parse a single file no matter the type (Form 990, Form 990-EZ, Schedule J, etc.). The function takes four arguments:

- tags is a data frame of the tags associated with the form you want to parse. For example, if you wanted to parse a directory of Schedule J files, the tags data frame would contain tags found in Schedule J files.
- file is the XML file that you would like to parse.
- form is the type of form you would like to parse. It can either be F990 or SCHED-A (the A can be replaced by another letter).
- repeatedTags is a list of tags that are repeated. This is useful for parsing Schedule J because we know that tags are repeated in this file.

parseFile checks if the repeatedTags list contains anything. If it does, then it looks at the tags in this list first and extracts data from the XML file for these specific tags. After extracting the repeated tags data, it looks at the remaining tags in the tags data frame and extracts the data associated with those tags. If the repeatedTags list is empty, then the parser goes straight to the tags data frame step.

There is also a main function that handles the logic of looping through the directory. This function goes through the directory and checks if the file contains the tax form you want to parse. If that form exists in the XML file, then we call the parseFile function. If the form doesn't exist, then we go to the next file. The main function also prints out the total time it takes to parse a directory. This helps in determining how much time needs to be allocated when a job is run for future parsing.

8.3.4 RunPyScript.sh

A file with a '.sh' extension is a scripting file that contains commands to run [10]. Currently, RunPyScript.sh looks like Figure 9. The script allocates the number of nodes, the number of tasks per node, the number of CPUs per node, and the amount of time the job should run. At the end of the file is a couple Python commands to run any of the Python scripts in our repository. In Figure 9, we can see that part-01 of Form 990-PF from 2015 will be parsed, Schedule A will be parsed, and duplicates from NPTR.db will be deleted. This can be changed depending on which forms need to be parsed and if any duplicates need to be deleted.

```
#!/bin/bash
#SBATCH --account=nonprofit_tax_data
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=4
#SBATCH --cpus-per-task=32
#SBATCH --time=47:59:59

python ParseTaxData.py 2015 F990 PF 01
python ParseTaxData.py 2015 SCHED-A
python DeleteDuplicates.py
```

Figure 9: Content inside RunPyScript.sh

9 Lessons Learned

The following describes the original timeline and the new timeline, why the timeline changed, any problems I faced, how I solved the problems, and any future work for teams in a next semester.

9.1 Timeline

I initially created my timeline at the end of August 2022, but I had to revise it once an issue was found.

9.1.1 Original Timeline

My original timeline is shown in Table 5.

Milestone	Plans
M1 09/09/2022	Make initial contact with Dr. Zach, find a time and day to have weekly meetings
M2 09/23/2022	Understand what needs to be done and clean the data already in the database
M3 10/07/2022	Parse new tax data using the Schedule J files and code created from Spring 2022
M4 10/21/2022	Add new data using Schedule J files to the database created in Spring 2022 semester
M5 11/04/2022	Compute basic statistics on the Schedule J data
M6 11/18/2022	Create a machine learning algorithm to find tourism offices
M7 12/07/2022	Visualize the data

Table 5: Original Timeline

During the first few weeks of September, I was researching about Form 990 and the Schedules, to understand all the tax jargon involved with this project. Also, I was able to get access to the git repository with the code from the Spring 2022 semester and the ARC cluster that stores the database created last semester. I looked through the code and noticed an issue immediately. The code wasn't documented well, so I didn't really know what each file did. I looked at the prior team's report to see if that clarified anything, but there was minimal explanation on the code there as well. I went ahead and created a README to the best of my ability to discuss what each file in the git repository does.

After I became familiar with the project, Dr. Zach wanted me to extract some information from the database created last semester. He needed compensation data for the years 2018 - 2020. When I was extracting the information, I noticed it took a significant amount of time extracting data from each table in the database. Also, I noticed that there were a lot of columns in each of the tables. I brought this issue up with with Dr. Zach in one of our weekly meetings and Dr. Zach realized that the database created in Spring 2022 was not what he wanted. This was around the end of September and this is when the timeline had to be revised.

9.1.2 Revised Timeline

Table 6 was created after we discovered the issue in the original database.

Milestone	Plans
M1 09/09/2022	Make initial contact with Dr. Zach, find a time and day to have weekly meetings
M2 09/23/2022	Understand what needs to be done and clean the data already in the database
M3 10/07/2022	Parse a single Schedule J file
M4 10/21/2022	Parse a directory of Schedule J files
M5 11/04/2022	Test the parser on other Form 990
M6 11/18/2022	Add the Schedule J data and Form 990 data to a new database and visualize the data in a meaningful way
M7 12/07/2022	Document the code and git repository

Table 6: Revised Timeline

The first two milestones remained the same, but after that the schedule was changed in order to create the universal parser.

9.2 Problems

There were three main problems that occurred in this project.

1. During the project, the original objectives had to be changed due to an issue in the previous group's work. When looking at the database the group from Spring 2022 created, eo.db, it was discovered that they had hundreds of columns for each table. After looking at some of the columns, Dr. Zach and I came to the conclusion that they didn't map the various tags correctly to a universal tag. As mentioned in section 5, there could be multiple tags that store the same information; their format is just slightly different. For example, TaxYear and TaxYr are the same thing with slight changes in their name.

2. The code created in Spring 2022 was not well-documented. There were minimal comments in the Jupyter Notebooks and Python scripts. Also, there was no README that gave instructions on how to run the files.

3. The directories that contained the XML files seemed to have duplicate files. When I created visualizations from the data that was extracted, I found there were some duplicates in the data.

9.3 Solutions

In order to combat the problems that occurred, Dr. Zach and I came to the following conclusions:

1. We would have to change the scope of this project and now create a universal parser that can parse Form 990 and the schedules. Also, the new database needs to have universal columns for each form. For example, Schedule J will have a table for each year and all those tables will have the same universal columns. Something similar will be done for Form 990.

2. I should ensure by the end of the semester that the Jupyter Notebook and Python scripts are well-documented. Additionally, I should make sure the README is detailed and gives instructions on any requirements, and how to run the files.

3. I need to create a Python script that will go through the NPTR database and delete any duplicate rows. Once the parser is run, then another Python script will be executed that would delete the duplicate rows in the SQLite tables.

9.4 Future Work

Since this research is in its early stages, there are many areas of improvement. However, there are 3 things that would help Dr. Zach significantly.

1. Add the remaining schedules data into the database. Dr. Zach and I were able to add data from Form 990, Form 990-EZ, and Schedule J into the database. However, I didn't have time to add all the data.

2. Create a website that allows users to search for specific information from the XML files. One example is: searching for a nonprofit organization and getting their Schedule J information from the 2015 tax year.

3. Develop a machine learning algorithm that can predict which nonprofit organizations are tourism offices. The database that has been created includes all nonprofit organizations, but it is unknown which is a tourism office. The only way you know which is a tourism office is if you know the Employer Identification Number (EIN) for the tourism office. Creating a machine learning algorithm to predict the tourism offices would save Dr. Zach from having to find the hundreds of EIN numbers, for every tourism office in the United States.

10 Acknowledgements

I would like to thank Dr. Zach for proposing this project and working with me to create a useful database. Additionally, I would like to thank Huihui Zhang, a Ph.D. student in the Pamplin College of Business, for her help with ARC. Lastly, I like to thank Dr. Fox for his guidance throughout the semester.

11 References

[1] Rettig, Charles. "E-File for Charities and Non-Profits." Internal Revenue Service, IRS, 26 Aug. 2022, https://www.irs.gov/e-file-providers/e-file-for-charities-and-non-profits. Web. Accessed 14 Nov. 2022.

[2] Rettig, Charles. "Gross Receipts Defined." Internal Revenue Service, IRS, 20 May 2022, https://www.irs.gov/charities-non-profits/gross-receipts-defined#:~:text=Gross%20receipts%20are%20the%20total,subtracting%20any%20costs%20or%20expenses. Web. Accessed 14 Nov. 2022.

[3] Sundaram, Agie, and Naga Palanisamy. "Schedules for Form 990: E-File 990 Schedules." ExpressTaxExempt, 2022, https://www.expresstaxexempt.com/form-990-schedules/. Web. Accessed 14 Nov. 2022.

[4] Rettig, Charles. "What's the Difference between Form 990-N and 990-Ez?: File 990." File 990, 25 June 2019, https://www.file990.org/blog/2019/06/25/568#:~:text=Form%20990%2DEZ%20is%20the,annual%20gross%20receipts%20over%20%24200%2C000. Web. Accessed 14 Nov. 2022.

[5] Howson, Steph. "Python XML Tutorial: Element Tree Parse & Read." DataCamp, DataCamp, 6 Mar. 2018, https://www.datacamp.com/tutorial/python-xml-elementtree. Web. Accessed 3 Oct. 2022.

[6] Richter, Stephan. "XML and HTML with Python." Lxml - XML and HTML with Python, Lxml, 9 July 2022, https://lxml.de/. Web. Accessed 3 Oct. 2022.

[7] Willems, Karlijn. "Pandas Tutorial: DataFrames in Python." DataCamp, DataCamp, 14 Jan. 2019, https://www.datacamp.com/tutorial/pandas-tutorial-dataframe-python. Web. Accessed 1 Sept. 2022.

[8] Herdman, Terry, and Kevin Shinpaugh. "About ARC." Advanced Research Computing — Virginia Tech, Virginia Tech, 31 Jan. 2022, https://arc.vt.edu/about.html. Web. Accessed 4 Dec. 2022.

[9] Zaporozhets, Dmytro, and Sytse "Sid" Sijbrandij. "The One DevOps Platform." GitLab, GitLab Inc, 2014, https://about.gitlab.com/. Web. Accessed 4 Dec. 2022.

[10] Arrows, Kevin. "What Are SH Files and How to Execute Them?" Appuals.com, Appuals, 13 Oct. 2022, https://appuals.com/what-are-sh-files-and-how-to-execute-them/. Web. Accessed 4 Dec. 2022.

[11] Barker, Kaleb, Jack Lacey, Long Nguyen, and Nicky Huynh. TaxData. CS4624 team term project submission, May 8, 2022, Blacksburg, Virginia, http://hdl.handle.net/10919/109974, accessed Dec. 4, 2022.