U.S. State Tourism

Final Report

Danya Shere, Rebecca Mueller, Ahmad Ayub, Lexi Fabian, Akshat Shah

CS 4624: Multimedia, Hypertext, Information Access

Virginia Tech

Blacksburg, VA 24061

May 11, 2020

Instructor: Dr. Edward A. Fox

Client: Dr. Florian Zach

Table of Contents

Table of Figures

Table of Tables

I. Executive Summary/Abstract

In the United States, every state has a tourism website. These sites highlight the main attractions of the state, travel tips, and blog posts among other relevant information. The funding for these websites often comes from occupancy taxes, a form of taxes that comes from tourists who stay in hotels and visit attractions. Therefore, current and past tourists fund the efforts to draw future tourists into the state.

Since state tourism is funded by the success of past tourism efforts, it is important for researchers to spend their time and resources on finding out what efforts were successful and which weren't. With this comes the importance of seeing trends in past tourism endeavors. By examining past tourism websites, patterns can be drawn about information that changed, from season to season and year to year. These patterns can be used to see what researchers deemed as successful tourism efforts, and help guide future state tourism decisions.

Our client, Dr. Florian Zach of the Howard Feiertag Department of Hospitality and Tourism Management, wants to use this historical analysis on state tourism information to help with his research on trends in state tourism website content. Iterations of the California state tourism website, among other sites, are stored as snapshots on the Internet Archive and can be accessed to see changes in websites over time. Our team was given Parquet files of these snapshots dating back to 2008. The goal of the project was to assist Dr. Zach by using the California state tourism website, visitcalifornia.com, and these snapshots as an avenue to explore data extraction and visualization techniques on tourism patterns to later be expanded to other states' tourism websites.

Python's Pandas library was utilized to examine and extract relevant pieces of data from the given Parquet files. Once the data was extracted, we used Python's Natural Language Processing Toolkit to remove non-English words, punctuation, and a set of unimportant "stop words". With this refined data, we were able to make visualizations regarding the frequency of words in the headers and body of the website snapshots. The data was examined in its entirety as well as in groups of seasons and years. Microsoft Excel functions were utilized to examine and visualize the data in these formats.

These data extraction and visualization techniques that we became familiar with will be passed down to a future team. The research on state tourism site information can be expanded to different metadata sets and to other states.

II. Introduction

The goal of this project is to utilize extracted data from an archive of the California state tourism website, visitcalifornia.com, to make meaningful visualizations on changes over time. We hope to set the groundwork for future teams to apply our extraction and visualization techniques to other state tourism websites so that more research can be done on state tourism efforts. We are working on this project for our client, Dr. Florian Zach of the Howard Feiertag Department of Hospitality and Tourism Management, as a part of the CS 4624: Multimedia, Hypertext, and Information Access capstone course taught by Dr. Edward Fox.

Snapshots of the California state tourism website were taken and stored by the Internet Archive. These snapshots are of various pages of the website on various dates in the years 2008-2019. From this resource, we were given data to work with in the Parquet [2] file format. These Parquet files contain the timestamp, URL, and HTML file, among other things, of each snapshot of the site. Python's Pandas data analysis library [3] was used to open these Parquet files and extract relevant information from them.

Once this information was extracted, Python's Natural Language Toolkit, NLTK [4], was used to remove unnecessary punctuation and non-English languages from the text. This version of the data was then used to experiment with visualizations based on different metadata.

We organized our extracted data, refined NLTK files, and visualizations into folders with clear documentation to be passed on to future groups. The intention is that other teams can expand on and utilize the techniques we experimented with to continue the research on state tourism websites. We hope our work can serve as a baseline to expand research to other states and metadata sets.

A. Team Roles [Table 1]

Team Member	**Role(s)**
Danya Shere	Project Lead, Data Visualization
Rebecca Mueller	Data Visualization
Ahmad Ayub	Data Extraction
Lexi Fabian	Natural Language Processing
Akshat Shah	Data Extraction

III. Requirements

The main requirement of this project is to use snapshots of the California state tourism website, visitcalifornia.com, dating back to 2008, to experiment with data extraction techniques for retrieving varied metadata and visualization techniques to display the information we find. Once these methods are effectively working for the California data, it is our hope that future project teams will apply them to other state tourism websites so research on state tourism patterns can be continued.

A. Data Extraction and Refactoring

The data for visitcalifornia.com was given to us in the form of four Parquet [2] files. We were to take these files and extract relevant information from them. This included extracting the text from the HTML file and structuring it to be more readable. To make the text more readable, the body was separated from the header and footer and used for analysis. We also extracted the headers of each page represented in the snapshots. The headers included article titles and page titles. These were important in understanding what main topics were being presented on the site at a given time.

For both of these metadata sets, we are using the Python Natural Language Toolkit, NLTK [4], to refactor. NLTK can be used to remove non-ASCII characters and unnecessary punctuation, leaving us with just English text. We also compiled a list of “stop words” which were smaller, common English words that were unimportant in data analysis. These words are common enough that they would take away from the relevant terms that could be used for research. The refactoring of this data is necessary because it takes the information from complicated and lengthy Parquet files and makes it easy to understand and analyze. Through these methods, four Parquet files, containing over 50,000 snapshots each, were condensed into one file with meaningful pieces of data in it.

B. Visualizations

The visualizations of this data should aid analysis by researchers. Graphs and charts on metadata like frequency of keywords and length of HTML text are made to visualize the extracted information.

The refactored header information was analyzed by finding the frequency of words. This frequency was found on the entire refactored dataset but also on smaller pieces of data. The data was separated by season to see if there were trends based on the time of year as well as in groups of years to see if overtime, relevant topics changed. These visualizations helped us begin to see patterns in state tourism site changes.

IV. Implementation

A. Choosing to use Jupyter Notebooks

Our client, Dr. Zach, had recommended using Jupyter Notebooks [5] for this project because of prior experience he had with the tool. One of the team members also had experience with the tool, which made setup easier for everyone else. Jupyter provides what is basically an IDE with a much easier to read output system. Jupyter allows us to have outputs for code snippets intermingled throughout the file so that certain snippets of the code can be run instead of the entire file at once, and only those snippets that you want to provide outputs can do that. This makes things convenient since we’re working with data in tabular format, and we can see what our data looks like every time we change it. It was also helpful for the team members that hadn’t used Jupyter before to familiarize themselves with a new tool.

B. Python tools

1. Pandas - Using Python’s “pandas” [3] made sense because it’s a very common tool for working with data analysis. It was also recommended by Dr. Zach, and a couple of the team members already had experience with the library. The Dataframe objects in Pandas allow easy manipulation of data in tabular format, which is how the data from the previous project (Doan 2019) was organized. The Dataframes are used once to filter information from the Parquet files into CSV files and then used again to read the CSV files to perform natural language processing on the data.

2. BeautifulSoup - The “BeautifulSoup” [6] package in Python was also used to read the HTML files provided for each snapshot from the Internet Archive.

3. NLTK [4] - In one of our last client meetings, Dr. Zach suggested to think about using the Natural Language Toolkit in Python to clean up our extracted data. At the time, the data was a bit unorganized and hard to read. After researching what the toolkit has to offer, we agreed that this would be an efficient way to clean up the headers by removing punctuation marks and non-English words.

C. Visualization Techniques

1. Microsoft Excel - While experimenting with the data in its early stages, Excel [10] proved to be a helpful tool because the header and timestamp information was extracted as a CSV file that could be nicely imported into the program. Excel VBA [9] functions were utilized to experiment with this extracted and refined data. These are built-in and user-defined functions that are used in Excel’s programming environment called Virtual Basic for Applications (VBA). A frequency function, FreqWords [11], and a sort function, SORT [12], were used to find the frequency of words in the headers and body of the text and sort them by most frequent. Then, Excel’s built in graph features were utilized to display these words and their counts.

We found that Excel did not take well to large amounts of data. For example, when the FreqWords function was run on a data set, errors would prevent the execution on sets greater than 30,000 columns. To get around this issue, the set was split into smaller parts and the function was run on the sections separately. Once this was completed, the resulting arrays of word frequencies were combined.

Because of these size constraint issues, we don't think that Excel is the most optimal visualization and analysis tool. It was useful to be able to quickly review the data but for future endeavors, other tools should be researched.

2. Pandas visualization tools - Pandas visualization tools provide many plotting libraries with lots of different features to visualize the high-dimensional data. Some libraries are Matplotlib, Seaborn, Bokeh and Plotly (create Interactive plots). These plotting libraries can be used to create highly customized plots such as scatterplot, distribution plots (Box Plots, Histogram), Heatmaps, and Parallel Coordinates. Due to time constraints, these tools were researched but not utilized. However, we believe future teams will find these tools very useful, especially with larger datasets. Since the data was extracted using Pandas, using their visualization tools as well should be a useful integration.

V. User's Manual

A. Use Cases/Tasks Supported

The users of our system include our client Dr. Florian Zach, state tourism researchers, and student teams who continue our work in the future. Our data files, code, and visualization are organized in a way that all of these user groups can easily navigate through our work to use it for their respective purposes.

Dr. Zach and other tourism researchers will be able to look through the graphs and tables that were created to aid their understanding of patterns throughout the years of state tourism websites. The frequency of words used on these sites will give these researchers a look into what topics were relevant to display at different time periods. This frequency analysis across seasons can be used to understand what attractions are advertised depending on the weather and time of year. Then, the frequency analysis throughout the years can be used to see which topics become more or less relevant as time passed.

For the future student teams, our files can form as a research tool. For the expansion of this project to other states and forms of metadata, the work we did constitute a good baseline. Our written code for data extraction and language processing will speed up the process for those preliminary steps in the future. Therefore, more focus can be put on analyzing the data and visualizing it. Our folders are compartmentalized in a way that it is very clear what code and files are where. This is explained in the following section.

B. File Interpretation

Our system included data files, data extraction code, NLTK [4] code, and visualizations. The data files used in this project included 4 Parquet files with all the original data in them, CSV files containing the header and timestamp information, a CSV file containing the HTML body text, and the language processed versions of these files. The data extraction code was written to be able to extract the headers, timestamps, and URLs of each snapshot in the Parquet files and to extract the text from the HTML files and separate the body of this text. The NLTK code was used to remove non-English words, unnecessary punctuation, and unimportant “stop words” from the data sets. Finally, the visualization files utilize Excel VBA functions [9] and graphing tools to create tables and graphs of the metadata.

All of these sections are separated into folders and each file in these folders has representative filenames. This will make it very easy for our client and state tourism researchers to understand where the data and visualizations are and for future student teams to understand where various code is. This compartmentalization was important for us to spend time on because of the nature of our project being used as a research tool and the importance of being able to pass down our work to future teams.

VI. Developer's Manual

Our Git repository is linked here:

https://github.com/USStateTourismWebsites/Data-Extraction

A. Jupyter Installation

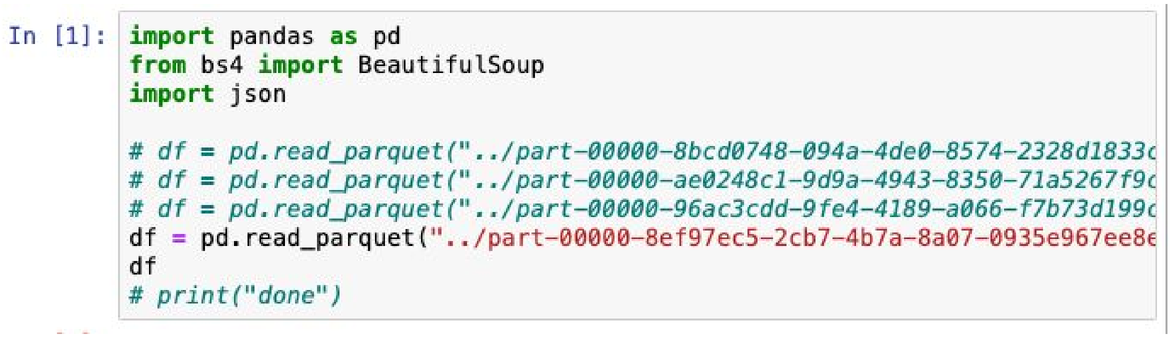
To use Jupyter [5], you need to download Anaconda Navigator [7], which is a data science platform for Python. Once you have downloaded Anaconda and run the application, you can launch the Jupyter Notebook platform which will open in a new browser window. You will see the same File Manager in the Workbench. Now, at the top right of the files tab, click the New button and select Python3 as your kernel environment to create a new notebook. A new tab launches in the browser window with the file name “Untitled”. You would have to install some packages before you run the Python script file/notebook. In the cell, simply input:

```
pip install pyarrow
```

Run the cell. Once all the packages are installed, you need to close the notebook and restart the kernel (Jupyter Notebook). You will have to download the Python script file and data files from the project folder. After you restart the kernel, you navigate through the file manager where you downloaded the file and open it. Now, run the notebook and wait for the process to finish.

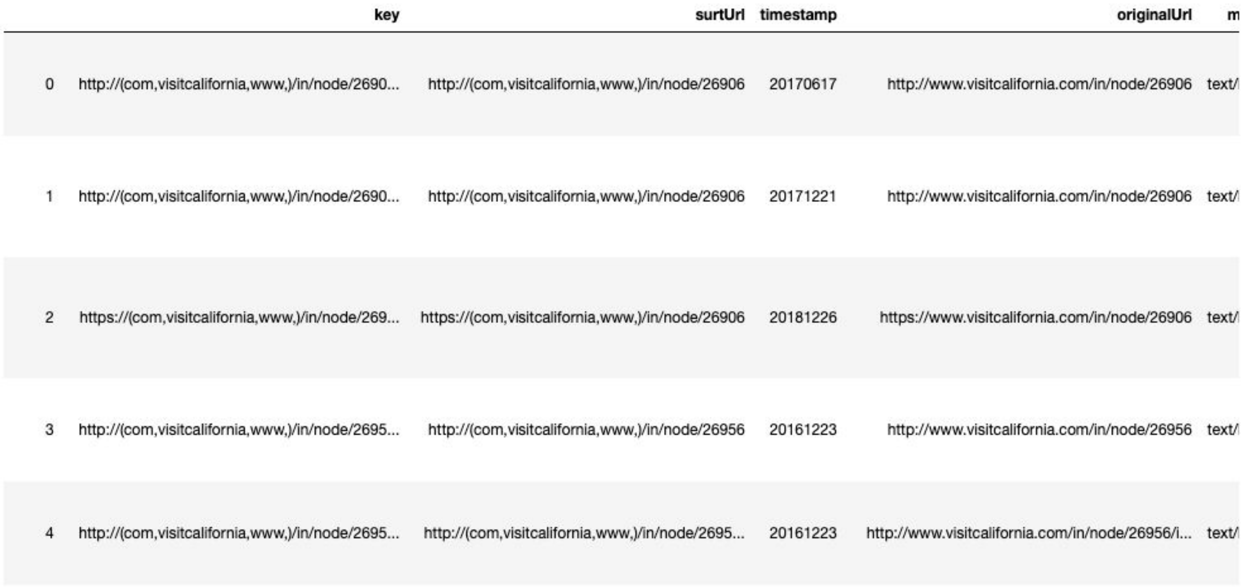
B. Organizing headers, timestamps, and URLs for website snapshots

The notebook titled “collectHeaderTimestampUrl” was used to read through the Parquet files and reorganize the headers for each snapshot alongside the timestamp and the URL from the snapshot. The notebook starts out by reading the Parquet files using the pandas [3] library as a Dataframe [8], which is a table-style data structure in pandas. Figure 1 shows the line of code that reads the Parquet file as a Dataframe, and Figure 2 shows what the output looks like in Jupyter.. This particular Parquet file has 48,809 rows and 14 columns, so not all of them are shown in Figure 2’s screenshot. This file takes about 65 seconds to be read into a Dataframe, although sometimes that can take up to twice as long depending on the file size.



```
In [1]:  import pandas as pd
         from bs4 import BeautifulSoup
         import json

         # df = pd.read_parquet("../part-00000-8bcd0748-094a-4de0-8574-2328d1833c
         # df = pd.read_parquet("../part-00000-ae0248c1-9d9a-4943-8350-71a5267f9c
         # df = pd.read_parquet("../part-00000-96ac3cdd-9fe4-4189-a066-f7b73d199c
         df = pd.read_parquet("../part-00000-8ef97ec5-2cb7-4b7a-8a07-0935e967ee8e
         df
         # print("done")
```

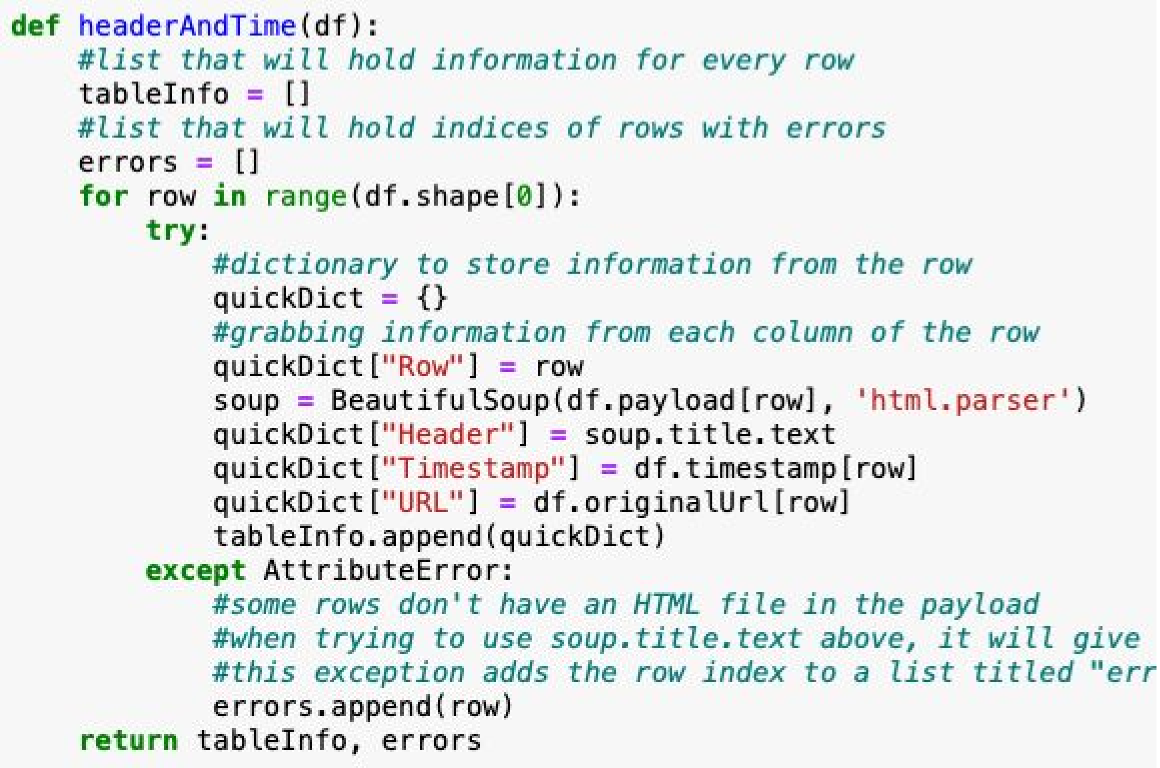
[Figure 1] The line of code using Pandas to load the Parquet file into a Dataframe



	key	surtUrl	timestamp	originalUrl	m
0	http://(com,visitcalifornia,www,)/in/node/2690...	http://(com,visitcalifornia,www,)/in/node/26906	20170617	http://www.visitcalifornia.com/in/node/26906	text/
1	http://(com,visitcalifornia,www,)/in/node/2690...	http://(com,visitcalifornia,www,)/in/node/26906	20171221	http://www.visitcalifornia.com/in/node/26906	text/
2	https://(com,visitcalifornia,www,)/in/node/269...	https://(com,visitcalifornia,www,)/in/node/26906	20181226	https://www.visitcalifornia.com/in/node/26906	text/
3	http://(com,visitcalifornia,www,)/in/node/2695...	http://(com,visitcalifornia,www,)/in/node/26956	20161223	http://www.visitcalifornia.com/in/node/26956	text/
4	http://(com,visitcalifornia,www,)/in/node/2695...	http://(com,visitcalifornia,www,)/in/node/2695...	20161223	http://www.visitcalifornia.com/in/node/26956/i...	text/

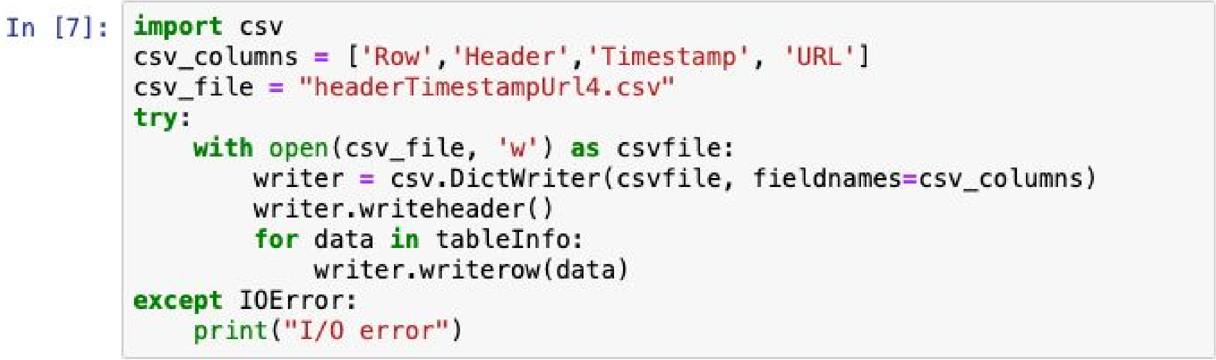
[Figure 2] The Dataframe produced by reading one of the Parquet files in a Jupyter Notebook

The “collectHeaderTimestampUrl.ipynb” file (shown in Figure 3) contains a method named “headerAndTime” that runs through a Dataframe going through each row, grabbing the piece of data from the columns that contain the HTML file for each snapshot, as well as the timestamp and URL. The method uses the BeautifulSoup package to read the HTML file and grab the header. Each row is put into a dictionary, alongside the row number that it’s from, and then each row is appended to a list named tableInfo that holds the pieces of data grabbed from every row. All of the data collected when the method is returned is put into a CSV file (Figure 4, Figure 5) with a column for each piece of information. The headerAndTime method takes about two hours to run per Parquet file.



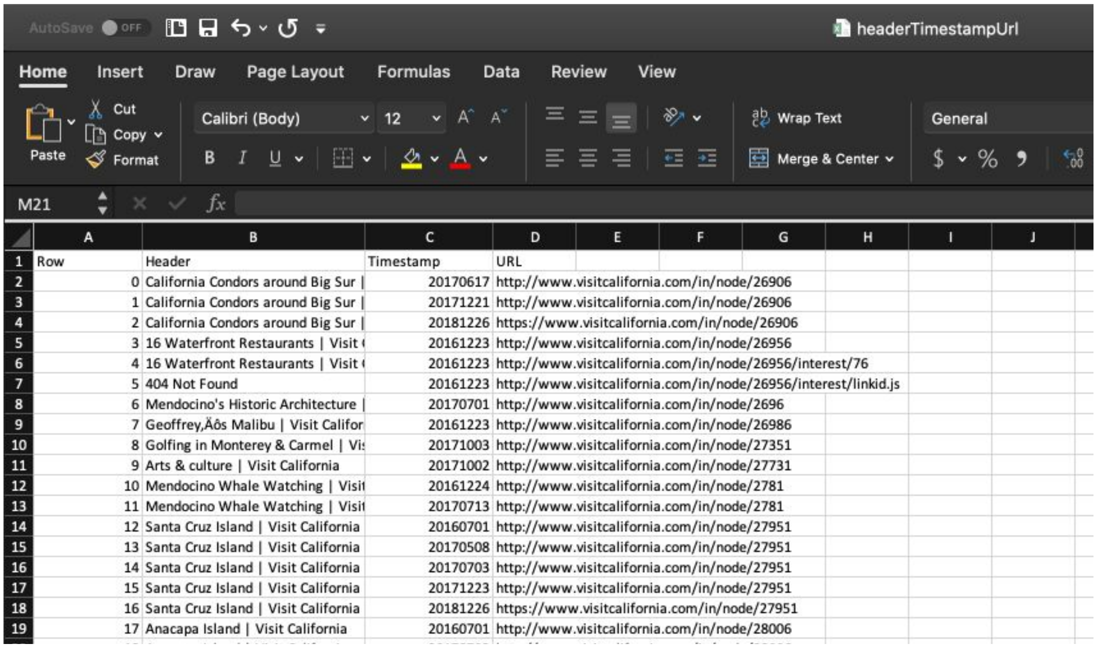
```
def headerAndTime(df):
    #list that will hold information for every row
    tableInfo = []
    #list that will hold indices of rows with errors
    errors = []
    for row in range(df.shape[0]):
        try:
            #dictionary to store information from the row
            quickDict = {}
            #grabbing information from each column of the row
            quickDict["Row"] = row
            soup = BeautifulSoup(df.payload[row], 'html.parser')
            quickDict["Header"] = soup.title.text
            quickDict["Timestamp"] = df.timestamp[row]
            quickDict["URL"] = df.originalUrl[row]
            tableInfo.append(quickDict)
        except AttributeError:
            #some rows don't have an HTML file in the payload
            #when trying to use soup.title.text above, it will give
            #this exception adds the row index to a list titled "err
            errors.append(row)
    return tableInfo, errors
```

[Figure 3] The headerAndTime function that picks pieces of information from the Dataframe and puts it into a format ready to insert into a CSV file



```
In [7]: import csv
        csv_columns = ['Row','Header','Timestamp', 'URL']
        csv_file = "headerTimestampUrl4.csv"
        try:
            with open(csv_file, 'w') as csvfile:
                writer = csv.DictWriter(csvfile, fieldnames=csv_columns)
                writer.writeheader()
                for data in tableInfo:
                    writer.writerow(data)
        except IOError:
            print("I/O error")
```

[Figure 4] Entering the output of the headerAndTime function into a CSV file

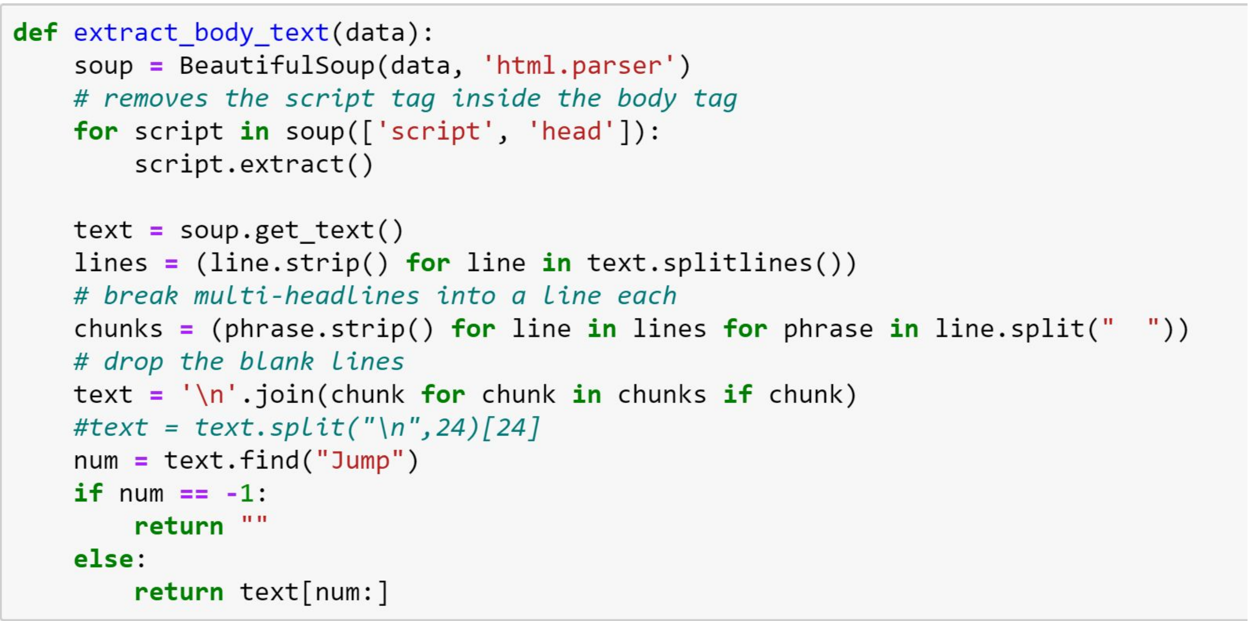


	A	B	C	D
1	Row	Header	Timestamp	URL
2	0	California Condors around Big Sur |	20170617	http://www.visitcalifornia.com/in/node/26906
3	1	California Condors around Big Sur |	20171221	http://www.visitcalifornia.com/in/node/26906
4	2	California Condors around Big Sur |	20181226	https://www.visitcalifornia.com/in/node/26906
5	3	16 Waterfront Restaurants | Visit (	20161223	http://www.visitcalifornia.com/in/node/26956
6	4	16 Waterfront Restaurants | Visit (	20161223	http://www.visitcalifornia.com/in/node/26956/interest/76
7	5	404 Not Found	20161223	http://www.visitcalifornia.com/in/node/26956/interest/linkid.js
8	6	Mendocino's Historic Architecture |	20170701	http://www.visitcalifornia.com/in/node/2696
9	7	Geoffrey‚Äôs Malibu | Visit Califor	20161223	http://www.visitcalifornia.com/in/node/26986
10	8	Golfing in Monterey & Carmel | Vi:	20171003	http://www.visitcalifornia.com/in/node/27351
11	9	Arts & culture | Visit California	20171002	http://www.visitcalifornia.com/in/node/27731
12	10	Mendocino Whale Watching | Visit	20161224	http://www.visitcalifornia.com/in/node/2781
13	11	Mendocino Whale Watching | Visit	20170713	http://www.visitcalifornia.com/in/node/2781
14	12	Santa Cruz Island | Visit California	20160701	http://www.visitcalifornia.com/in/node/27951
15	13	Santa Cruz Island | Visit California	20170508	http://www.visitcalifornia.com/in/node/27951
16	14	Santa Cruz Island | Visit California	20170703	http://www.visitcalifornia.com/in/node/27951
17	15	Santa Cruz Island | Visit California	20171223	http://www.visitcalifornia.com/in/node/27951
18	16	Santa Cruz Island | Visit California	20181226	https://www.visitcalifornia.com/in/node/27951
19	17	Anacapa Island | Visit California	20160701	http://www.visitcalifornia.com/in/node/28006

[Figure 5] An example CSV file before any analysis has been done on it

C. Body and Header Extraction

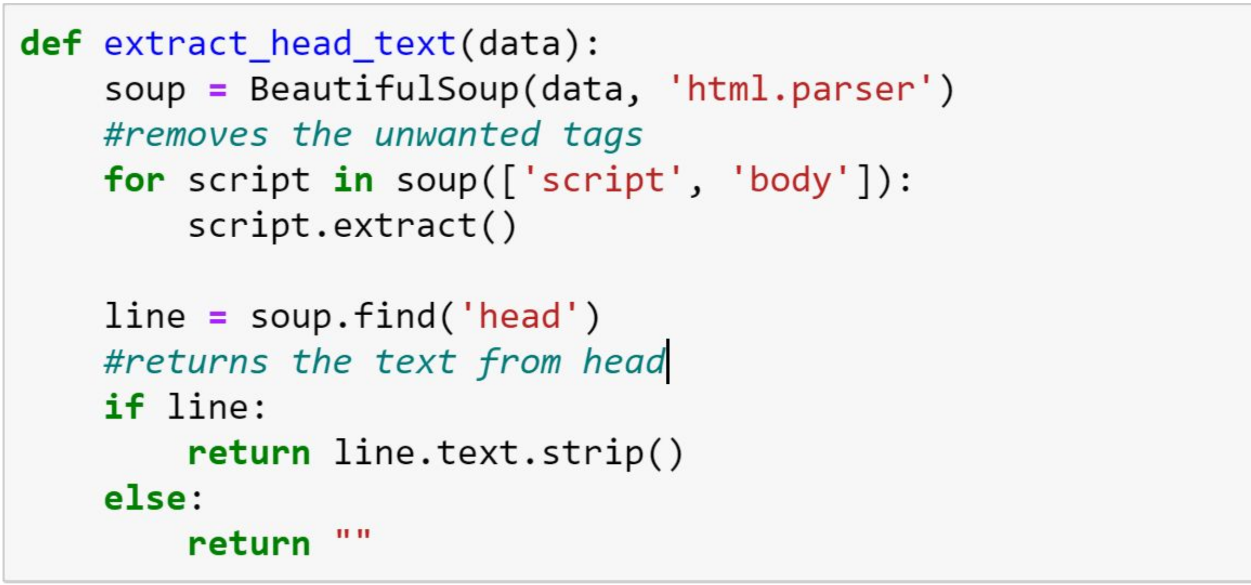
The file contains a method called “extract_body_text” which runs through the dataframe and extracts body text from the HTML file and uses the BeautifulSoup package to read the HTML text. It then runs through the HTML text and removes the script and head tag, so that we only have body tag left to extract data. The body text extracted from each row’s snapshot is split into lines and trimmed; further lines are split into words and trimmed as well. All the data is then joined together, but this data still contains some unwanted text which needs to be removed. The method then looks for a specific word and returns more refined data which can be used to do analysis.



```python
def extract_body_text(data):
    soup = BeautifulSoup(data, 'html.parser')
    # removes the script tag inside the body tag
    for script in soup(['script', 'head']):
        script.extract()

    text = soup.get_text()
    lines = (line.strip() for line in text.splitlines())
    # break multi-headlines into a line each
    chunks = (phrase.strip() for line in lines for phrase in line.split("  "))
    # drop the blank lines
    text = '\n'.join(chunk for chunk in chunks if chunk)
    #text = text.split("\n",24)[24]
    num = text.find("Jump")
    if num == -1:
        return ""
    else:
        return text[num:]
```

[Figure 6] The extract_body_text function that extracts body text from an HTML file and returns a list of data

The extract_head_text function works exactly like the extract_body_text function. It runs through the dataframe and extracts the text from the head tag in the HTML file. The method removes the script and body tag and then looks for the header tag which is used to return the trimmed text from the header tag.



```python
def extract_head_text(data):
    soup = BeautifulSoup(data, 'html.parser')
    #removes the unwanted tags
    for script in soup(['script', 'body']):
        script.extract()

    line = soup.find('head')
    #returns the text from head
    if line:
        return line.text.strip()
    else:
        return ""
```

[Figure 7] The extract_head_text function extracts header text from an HTML file

D. File Hierarchy

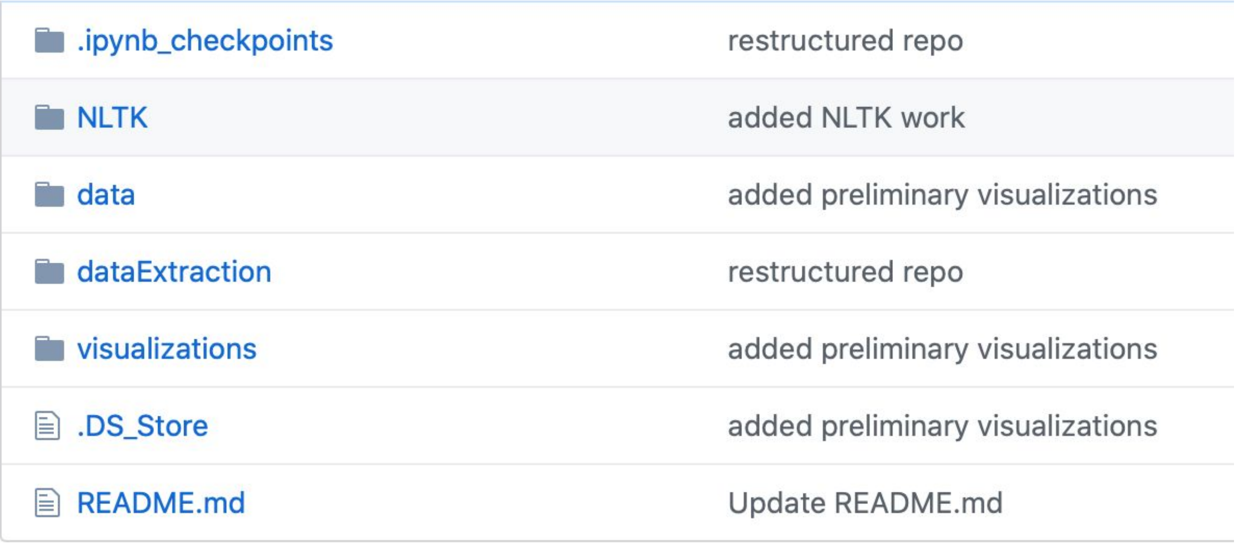
The project is organized into 4 folders. This structure is shown in Figure 8.

Data - The Data folder contains the 4 Parquet files with all the original data in them, CSV files containing the header and timestamp information, a CSV file containing the HTML body text, and the language processed versions of these files. These were all given representative filenames to differentiate so it should be easy to understand which data is in what file.

Data Extraction Code - The Data Extraction Code folder contains all the pandas code that was written. This includes the code that was written to extract the page headers, timestamps, and URLs as well as the code that was used to extract the text from the HTML files and separate the body.

NLTK Code - The NLTK Code folder contains all the code written for the Natural Language Processing Toolkit. This includes removing non-English words, unnecessary punctuation, and unimportant “stop-words”. This code was used on the extracted data to refactor it and make it more manageable to process.

Visualizations - The Visualizations folder contains the Excel files where the VBA functions were run and the actual visualizations themselves. The charts of the word frequencies are in the Excel files and the visuals are stored separately so they can be easily referenced. This folder also contains code that was written to assist in visualization and data analysis.



.ipynb_checkpoints	restructured repo
NLTK	added NLTK work
data	added preliminary visualizations
dataExtraction	restructured repo
visualizations	added preliminary visualizations
.DS_Store	added preliminary visualizations
README.md	Update README.md

[Figure 8] A snapshot of the Git repository

E. NLTK Installation

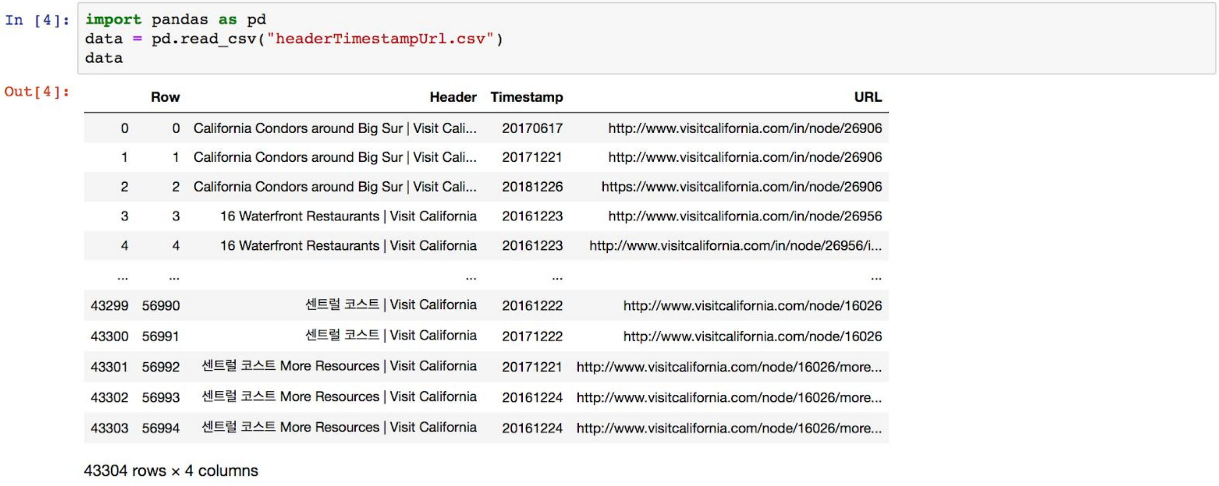
The Python programming language supports a series of libraries called the Natural Language Toolkit, or NLTK for short [4]. These packages allow users to manipulate and process data for many purposes. To utilize this feature, you must install NLTK on your machine. In your Juypter cell, type:

```
pip install --user -U nltk
import nltk
nltk.download()
```

Once the installation is complete, you are ready to put the toolkit to use. For this project, this feature becomes very useful for cleaning up the data extracted from the web archives.

F. NLTK Tokenization

An important feature of the NLTK is tokenization, which is the process of breaking up text into smaller parts. It is very similar to splitting up words in a string and putting them into an array. The notebook “NLTKFiltering” is used to filter out the punctuation included in the headers. The CSV file created in the collectHeaderTimestampUrl is read into a pandas Dataframe.



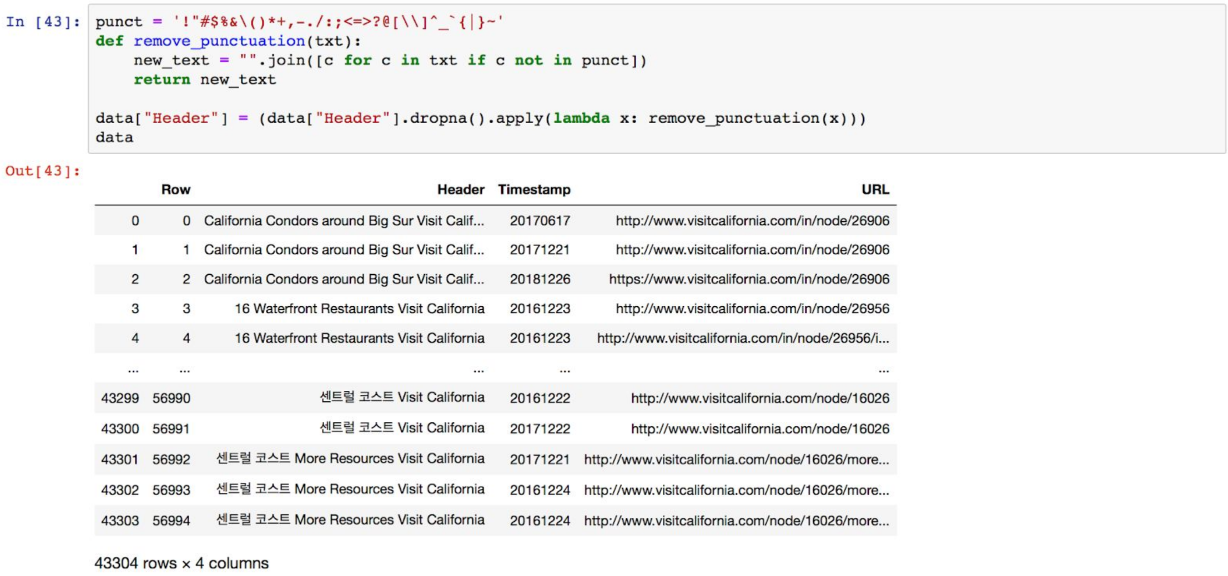
```
import pandas as pd
data = pd.read_csv("headerTimestampUrl.csv")
data
```

	Row	Header	Timestamp	URL
0	0	California Condors around Big Sur | Visit Cali...	20170617	http://www.visitcalifornia.com/in/node/26906
1	1	California Condors around Big Sur | Visit Cali...	20171221	http://www.visitcalifornia.com/in/node/26906
2	2	California Condors around Big Sur | Visit Cali...	20181226	https://www.visitcalifornia.com/in/node/26906
3	3	16 Waterfront Restaurants | Visit California	20161223	http://www.visitcalifornia.com/in/node/26956
4	4	16 Waterfront Restaurants | Visit California	20161223	http://www.visitcalifornia.com/in/node/26956/i...
...	...	...	...	...
43299	56990	센트럴 코스트 | Visit California	20161222	http://www.visitcalifornia.com/node/16026
43300	56991	센트럴 코스트 | Visit California	20171222	http://www.visitcalifornia.com/node/16026
43301	56992	센트럴 코스트 More Resources | Visit California	20171221	http://www.visitcalifornia.com/node/16026/more...
43302	56993	센트럴 코스트 More Resources | Visit California	20161224	http://www.visitcalifornia.com/node/16026/more...
43303	56994	센트럴 코스트 More Resources | Visit California	20161224	http://www.visitcalifornia.com/node/16026/more...

43304 rows × 4 columns

[Figure 9] How to read in the .csv file and put it into a Dataframe

Next, a variable called ‘punct’ is created to specify all punctuation that is to be removed. Since we do not want words with apostrophes to be split, that is the only punctuation mark not included in the list. The notebook includes a method named “remove_punctation()” which splits a string using tokenization, and removes any punctuation included in our list. The same approach is taken to remove non-ASCII characters from the headers. Multiple headers contain characters that are not in the English alphabet. The method “remove_non_ascii()” [13] was created to get rid of these words. After reviewing the filtered data, we realized that our data included many useless words such as “the”, “a”, “an”, “is”, etc. To remove these, we utilized the toolkit’s ‘Stop Words’ feature which is a list that includes words that can be ignored to filter out the meaningful data. To do this, tokenization splits a string into individual words, then removes all words that are included in the list. The result is a string containing only significant words that we want to analyze.

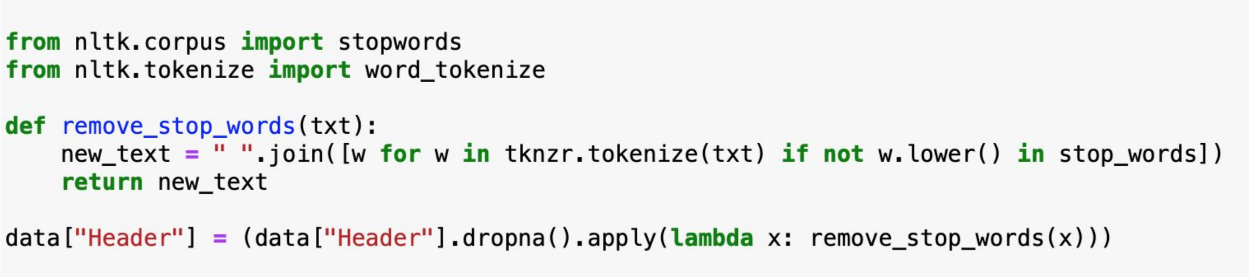


```
In [43]: punct = '!"#$%&\()*+,-./:;<=>?@[\\]^_`{|}~'
         def remove_punctuation(txt):
             new_text = "".join([c for c in txt if c not in punct])
             return new_text

         data["Header"] = (data["Header"].dropna().apply(lambda x: remove_punctuation(x)))
         data
Out[43]:
```

	Row	Header	Timestamp	URL
0	0	California Condors around Big Sur Visit Calif...	20170617	http://www.visitcalifornia.com/in/node/26906
1	1	California Condors around Big Sur Visit Calif...	20171221	http://www.visitcalifornia.com/in/node/26906
2	2	California Condors around Big Sur Visit Calif...	20181226	https://www.visitcalifornia.com/in/node/26906
3	3	16 Waterfront Restaurants Visit California	20161223	http://www.visitcalifornia.com/in/node/26956
4	4	16 Waterfront Restaurants Visit California	20161223	http://www.visitcalifornia.com/in/node/26956/i...
...	...	...	...	...
43299	56990	센트럴 코스트 Visit California	20161222	http://www.visitcalifornia.com/node/16026
43300	56991	센트럴 코스트 Visit California	20171222	http://www.visitcalifornia.com/node/16026
43301	56992	센트럴 코스트 More Resources Visit California	20171221	http://www.visitcalifornia.com/node/16026/more...
43302	56993	센트럴 코스트 More Resources Visit California	20161224	http://www.visitcalifornia.com/node/16026/more...
43303	56994	센트럴 코스트 More Resources Visit California	20161224	http://www.visitcalifornia.com/node/16026/more...

43304 rows × 4 columns

[Figure 10] The function remove_punct() that removes all punctuation from the Header data

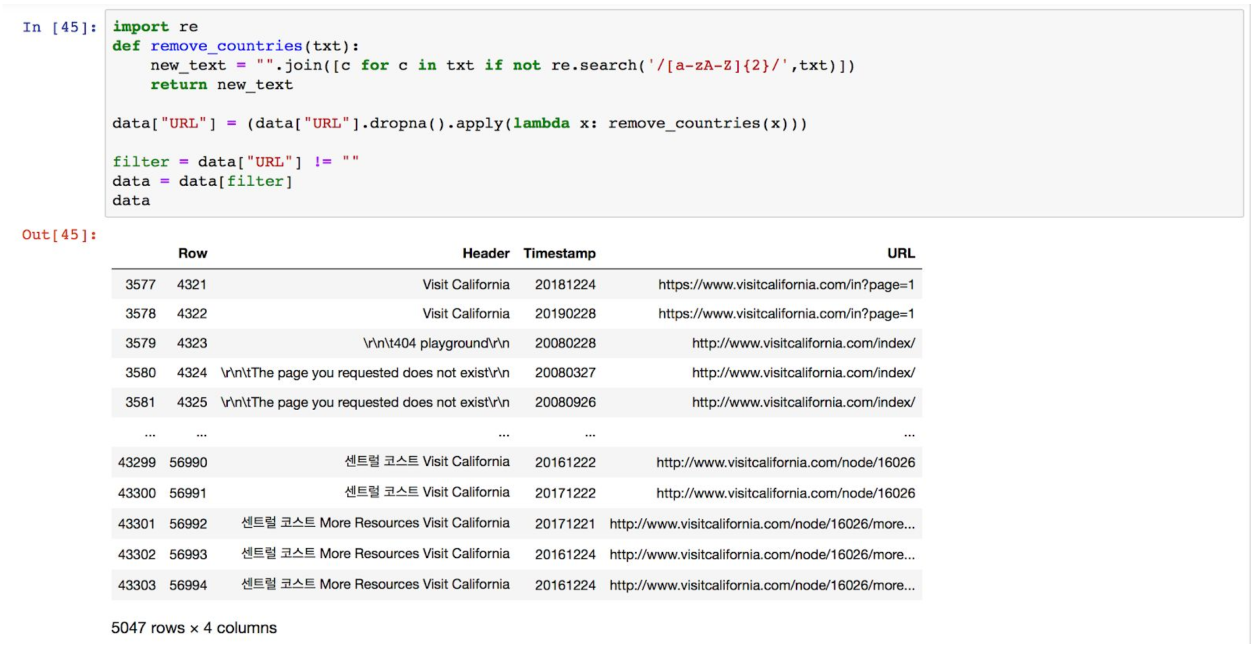


```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

def remove_stop_words(txt):
    new_text = " ".join([w for w in tknzr.tokenize(txt) if not w.lower() in stop_words])
    return new_text

data["Header"] = (data["Header"].dropna().apply(lambda x: remove_stop_words(x)))
```

[Figure 11] The function remove_stop_words() that removes all stop words from the Header data

G. Python Filtering

To remove rows where the URLs came from other countries, the Python Library 're' is imported to utilize its functions. Since foreign URLs include a two letter abbreviation in between two '/' characters, the method remove_countries() is created to delete these URLs. The "filter" method is then used to remove the affected rows. This minimizes the Dataframe from 43304 rows to 5074. The "filter" function is also very efficient for removing all unnecessary data including pages that could not be found.



In [45]:
```
import re
def remove_countries(txt):
    new_text = "".join([c for c in txt if not re.search('/[a-zA-Z]{2}/',txt)])
    return new_text

data["URL"] = (data["URL"].dropna().apply(lambda x: remove_countries(x)))

filter = data["URL"] != ""
data = data[filter]
data
```

Out[45]:

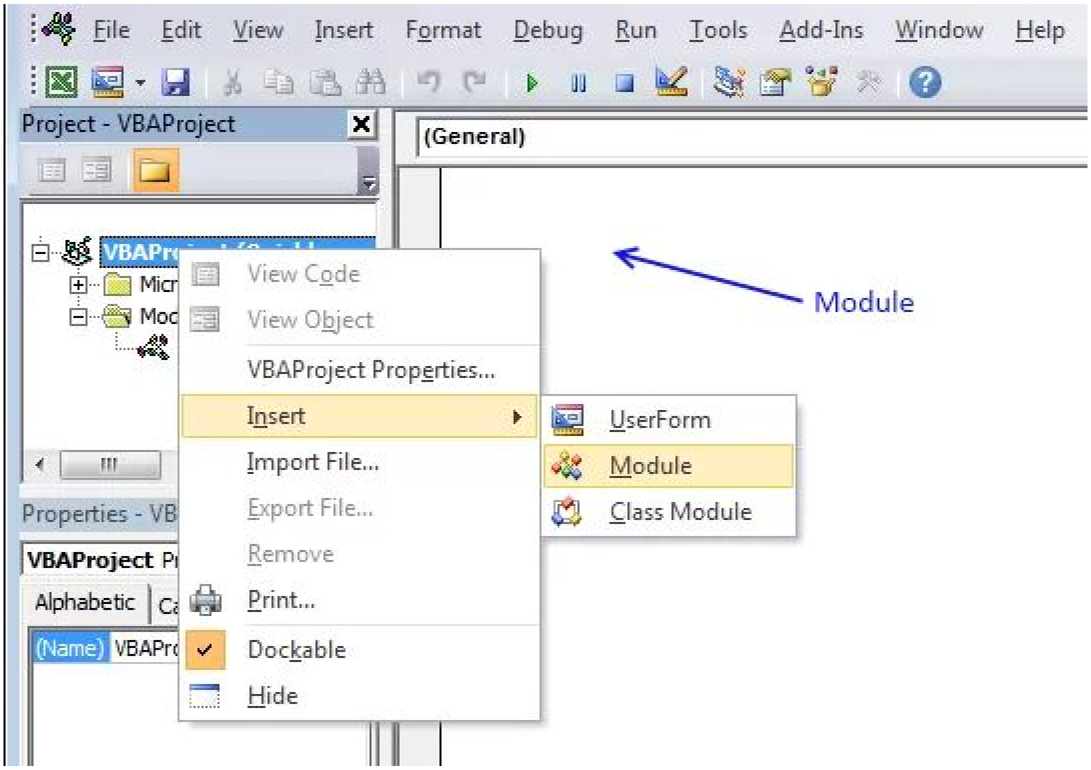
	Row	Header	Timestamp	URL
3577	4321	Visit California	20181224	https://www.visitcalifornia.com/in?page=1
3578	4322	Visit California	20190228	https://www.visitcalifornia.com/in?page=1
3579	4323	\r\n\t404 playground\r\n	20080228	http://www.visitcalifornia.com/index/
3580	4324	\r\n\tThe page you requested does not exist\r\n	20080327	http://www.visitcalifornia.com/index/
3581	4325	\r\n\tThe page you requested does not exist\r\n	20080926	http://www.visitcalifornia.com/index/
...	...	...	...	...
43299	56990	센트럴 코스트 Visit California	20161222	http://www.visitcalifornia.com/node/16026
43300	56991	센트럴 코스트 Visit California	20171222	http://www.visitcalifornia.com/node/16026
43301	56992	센트럴 코스트 More Resources Visit California	20171221	http://www.visitcalifornia.com/node/16026/more...
43302	56993	센트럴 코스트 More Resources Visit California	20161224	http://www.visitcalifornia.com/node/16026/more...
43303	56994	센트럴 코스트 More Resources Visit California	20161224	http://www.visitcalifornia.com/node/16026/more...

5047 rows × 4 columns

[Figure 12] The remove_countries() function that removes all URLs contain the substring “/../”

H. Excel VBA Installation

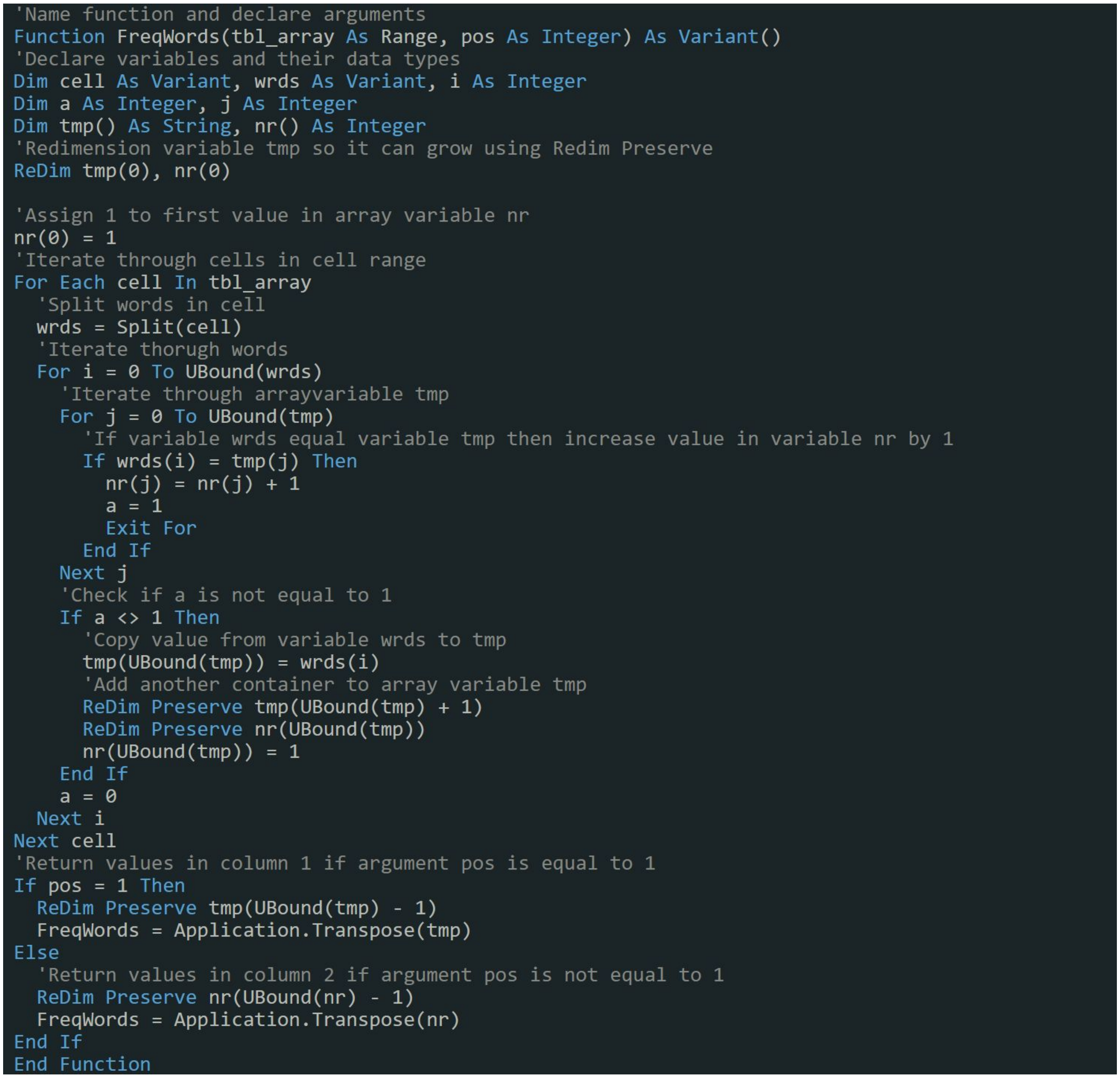
To add a new user defined function into your Excel workbook the Virtual Basic Editor is used. This can be accessed by pressing Alt-F11 while in Excel. In the Insert menu, click Module and write or paste the necessary VBA code into that editor. This process is shown in Figure 13. It is important to note that these functions can only be run on macro-enabled worksheets. So, files should be saved in the XLSM format instead of CSV.



[Figure 13] Opening up the VBA Editor

I. Excel FreqWords Function

The FreqWords function finds the frequency of all words in a data set. This was used on the headers of the pages after the NLTK code was run on it. It created arrays of the words and their respective frequencies in the order that they were found. The function took in a range of rows and a position. If the position was set to 1, this would print the array of words and if the position was set to any other number, it would print the word count. The code for this function is shown in Figure 14. It was pasted into the VBA Editor so it could be used.

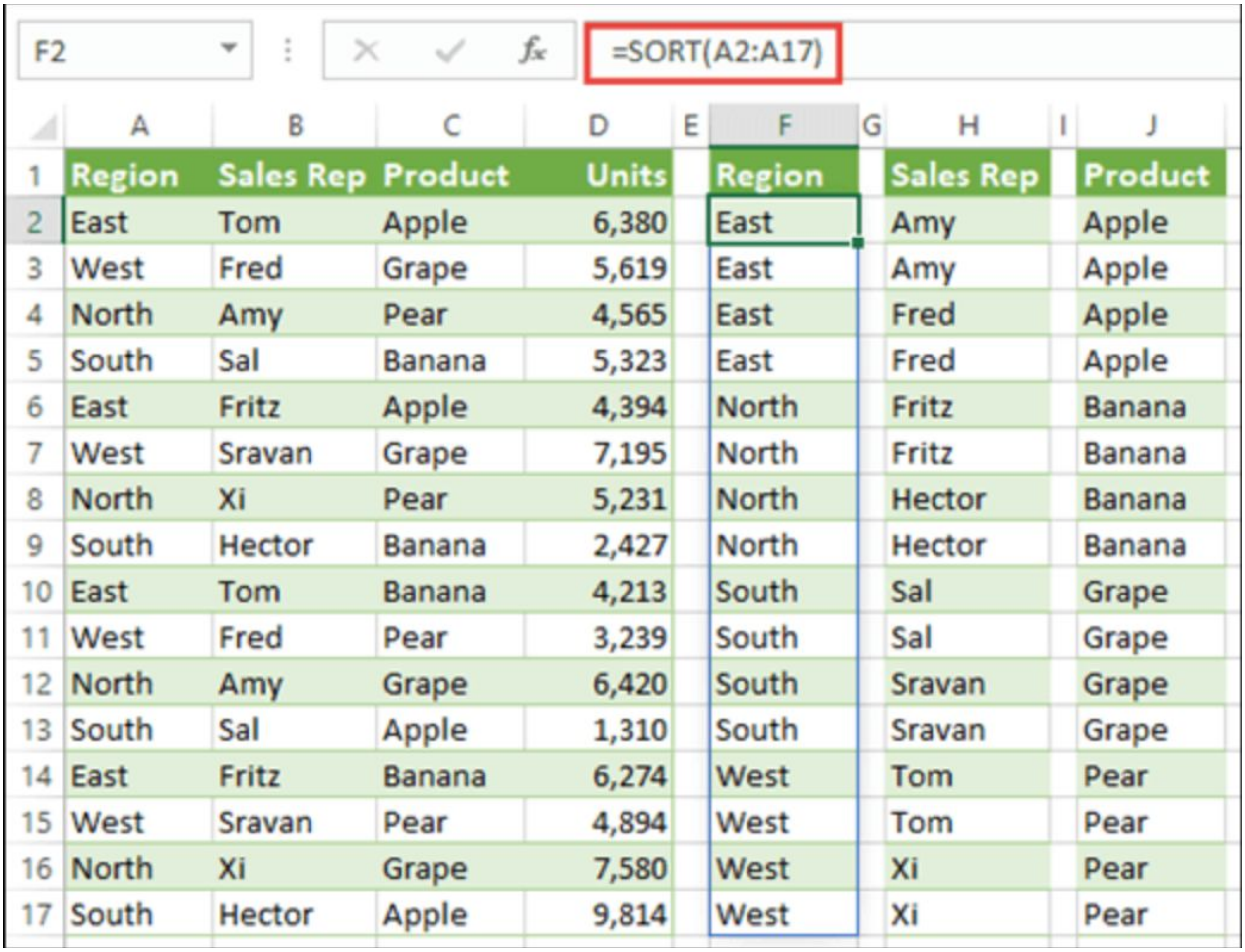


```
'Name function and declare arguments
Function FreqWords(tbl_array As Range, pos As Integer) As Variant()
'Declare variables and their data types
Dim cell As Variant, wrds As Variant, i As Integer
Dim a As Integer, j As Integer
Dim tmp() As String, nr() As Integer
'Redimension variable tmp so it can grow using Redim Preserve
ReDim tmp(0), nr(0)

'Assign 1 to first value in array variable nr
nr(0) = 1
'Iterate through cells in cell range
For Each cell In tbl_array
  'Split words in cell
  wrds = Split(cell)
  'Iterate thorugh words
  For i = 0 To UBound(wrds)
    'Iterate through arrayvariable tmp
    For j = 0 To UBound(tmp)
      'If variable wrds equal variable tmp then increase value in variable nr by 1
      If wrds(i) = tmp(j) Then
        nr(j) = nr(j) + 1
        a = 1
        Exit For
      End If
    Next j
    'Check if a is not equal to 1
    If a <> 1 Then
      'Copy value from variable wrds to tmp
      tmp(UBound(tmp)) = wrds(i)
      'Add another container to array variable tmp
      ReDim Preserve tmp(UBound(tmp) + 1)
      ReDim Preserve nr(UBound(tmp))
      nr(UBound(tmp)) = 1
    End If
    a = 0
  Next i
Next cell
'Return values in column 1 if argument pos is equal to 1
If pos = 1 Then
  ReDim Preserve tmp(UBound(tmp) - 1)
  FreqWords = Application.Transpose(tmp)
Else
  'Return values in column 2 if argument pos is not equal to 1
  ReDim Preserve nr(UBound(nr) - 1)
  FreqWords = Application.Transpose(nr)
End If
End Function
```

[Figure 14] VBA function FreqWords

J. Excel SORT function

The Excel SORT function [15] sorts the content of a range or an array. This function is built-in to Excel and does not need to be added like FreqWords did. It takes in a start and an end to a range and sorts it accordingly. This was used to sort the result of FreqWords so that it was in order of the most common to least common words in the dataset. An example of the SORT function can be seen in Figure 12.



F2

=SORT(A2:A17)

	A	B	C	D	E	F	G	H	I	J
1	Region	Sales Rep	Product	Units		Region		Sales Rep		Product
2	East	Tom	Apple	6,380		East		Amy		Apple
3	West	Fred	Grape	5,619		East		Amy		Apple
4	North	Amy	Pear	4,565		East		Fred		Apple
5	South	Sal	Banana	5,323		East		Fred		Apple
6	East	Fritz	Apple	4,394		North		Fritz		Banana
7	West	Sravan	Grape	7,195		North		Fritz		Banana
8	North	Xi	Pear	5,231		North		Hector		Banana
9	South	Hector	Banana	2,427		North		Hector		Banana
10	East	Tom	Banana	4,213		South		Sal		Grape
11	West	Fred	Pear	3,239		South		Sal		Grape
12	North	Amy	Grape	6,420		South		Sravan		Grape
13	South	Sal	Apple	1,310		South		Sravan		Grape
14	East	Fritz	Banana	6,274		West		Tom		Pear
15	West	Sravan	Pear	4,894		West		Tom		Pear
16	North	Xi	Grape	7,580		West		Xi		Pear
17	South	Hector	Apple	9,814		West		Xi		Pear

[Figure 15] Excel SORT function example

K. Filtering by season

Besides just analyzing all the data at once, we wanted to perform other analysis on it as well. A Java function was written to take the filtered headers and separate them by seasons given the timestamp. The function looks at one header at a time and determines which season it belongs to based on the date associated with it. It is then filtered to the appropriate season's CSV file. The code for this function is in the "Visualizations" folder of our repository in the file SeasonsAnalysis.java.

L. Combining arrays

The overall dataset and the dataset in the winter.csv file were too large to use the Excel FreqWords VBA function as is. Because of this, the frequency had to be found on halves of the datasets and later combined. A Java function was written to combine the frequency lists of these halves. The function creates a HashMap, appending to it when a new word is found and incrementing the frequency count when a duplicate word is found. This combines the halves into one list of words and their frequency counts. This can then be sorted and used for graphs and visualizations. The code for this can be found in the “Visualizations” folder of our repository in the file CombineValues.java.

M. Methodology

1. Goals of our users

    a. Dr. Florian Zach - This user’s goal is to be provided with user-friendly, filtered, and sorted data on state tourism information to help with his research on trends in state tourism website content. He also needs to be able to pass our code on to students in the future who will work on other state tourism websites. In addition, another user type is other state tourism researchers who may use our data and visualizations. Florian Zach and other researchers need to see our data in a visual way to see trends in the state tourism websites and make decisions about tourism efforts.

    b. Future project teams - They will have similar goals to Florian Zach in that they want user-friendly data, but they will want even more documentation to clarify all the work that we’ve done, since they don’t want to repeat anything we already tried. They also need documentation to know how to use what we’ve created.

2. Subtasks of our goal

    a. Goal #1: Get user-friendly, filtered, and sorted data and create visualizations (main goal of Florian Zach and other researchers)

        Sub-tasks:

        Load the data from previous semesters

        Filter the data

        Sort the data in a way that is meaningful

        Make visualizations from this sorted data

        Export the data

In order to accomplish the goal of getting user-friendly, filtered, and sorted data and creating visualizations, we first need to get the tourism website data from previous semesters. Then we need to filter the data, since a lot of the data had errors (404 not found) and snapshots in different languages. Once the data is cleaned up, we need to sort the data or make it sortable, so that we can look at different trends. This includes making the data sortable by date and by keyword. From this sorted data, we can make different visualizations (histograms, word frequency analysis, etc.). Finally, we can export the clean data and visualizations to a central location for future researchers and students.

b. Goal #2: Document our code and progress

Sub-tasks:

Write down what we try in one place

Store all of our work in a place that is accessible

Document what works

Comment the code

In order to accomplish the goal of documenting our code and progress, a lot of the work must be done as we go. First, we have to decide to write everything down in a central location. This also includes storing all code and documentation in a central location, such as a Google Drive. Then, as we progress, we must document what we try and what works. We also must write comments in our code as we go along. If we keep track of doing all this as we go along, then by the end of the semester, we will have fully documented our code and our progress.

3. Implementation-based services

a. Goal #1: Get user-friendly, filtered, and sorted data and create visualizations

In order to implement this goal, the first step is to install Anaconda and set up Jupyter Notebook. This is so we can easily read the input file and work with it. We'll be using Python due to it's many libraries for data analysis. The input data will be the .parquet files from the previous semester’s team. We are given these files from our client, Dr. Zach. We need to use the Pandas library to read the input files into Dataframes and pick out information from that. We also need to use BeautifulSoup to read the HTML that is contained in the input files, and NLTK to filter the information that we don’t need, like words in other languages and English filler words. We’ll be using Python’s Pandas libraries as well as tools in Microsoft Excel to visualize the data that we collect and filter from the input files. The output will be these visualizations.

b. Goal #2: Document our code and progress

In order to implement this goal, the entire team needs to agree on a central location for documentation. We agreed on using Google Drive to store all our files and documentation. From there, we need to have a process for note taking. We agreed on a note-taker for every meeting and made sure to document what we tried, what failed, and what worked. We also made sure to document what code we had or write down what we tried in the drive, even if it was unsuccessful. This ensures that we have the most detail so that no one repeats work that we already tried, and future users know how to use what we create.

4. Workflows

User → Goal 1 → Workflow 1

Workflow 1 = Service 1A + Service 1B + Service 1C + Service 1D + Service 1E

Service 1A: Load the data from previous semesters using Anaconda, Jupyter Notebook, and Python

Service 1B: Filter the data using Pandas

Service 1C: Sort the data in a way that is meaningful using Pandas and Microsoft Excel

Service 1D: Make visualizations from this sorted data using Pandas and Microsoft Excel

Service 1E: Export the data

User → Goal 2 → Workflow 2

Workflow 2 = Service 2A + Service 2B + Service 2C + Service 2D

Service 2A: Write down everything

Service 2B: Store all of our work in Google Drive

Service 2C: Document what we try, what doesn't work, and what works

Service 2D: Comment the code

VII. Lessons Learned

A. Timeline/schedule [Table 2]

1/27	First Client Meeting - discussed overview of project, technologies to use, and visualizations to look into
2/2	Project Description Due - finalized deliverables
2/2 - 2/10	Research Period
2/10	Second Client Meeting - narrowed down relevant technologies, discussed plan for extraction
2/12	Created GitHub repository, began extraction
2/18	First Presentation to Class
2/19	Project shift from Virginia data to California
2/19 - 3/2	Familiarized ourselves with California data, extracted headers/timestamps as well as raw text
3/2	Third Client Meeting, discussed ideas for visualizations with new California data
3/6	Created frequency table and graph of words in headers
3/7 - 3/22	Extended Spring Break
3/27	Fourth Client Meeting, check-in

3/27 - 4/6	Restructuring of data, use of natural language processing tools
4/2	Second Presentation to Class
4/6	Interim Report Due
4/6 - 4/10	Creating visualizations with restructured data
4/9	Fifth Client Meeting
4/10 - 4/24	Additional visualizations, stretch goal evaluation
4/23	Sixth Client Meeting
4/26	Final Report Initial Submission
4/27 - 5/6	Project Wrap-Up
4/30	Third/Final Presentation to Class
5/1	Final Client Meeting

B. Problems

1. Virginia Parquet files - The original files we were given from the work of previous semester's groups were data from blog.virginia.org instead of virginia.org. This blog site contained articles about tourism in Virginia, but not information from the actual tourism website. Our clients decided this blog site would not be useful for the scope of this project and that an alternative data set would be needed.

2. Ten years of data vs. twenty - Initially, the scope of this project included using two decades worth of data (1998-2018). This would give a large range to look into to see changes in tourism patterns. However, in the conversion process of the extracted data that previous semester's groups executed, there were errors. The data that was being used from the Internet Archive came as both ARC (older version) and WARC (newer version) files but only the WARC files were converted to the Parquet versions we were using. Therefore, any data using the older ARC file format were not converted and could not be used.

3. COVID-19 Emergency - In the midst of the COVID-19 pandemic, spring break was extended for a week and our classes were made online for the remainder of the semester. This extra week of no class, along with the period of adjusting to online work, pushed back the progress of the project. The research consortium, VTURCS, where our work was to be displayed, was also cancelled. Therefore, a poster displaying our team's work for the semester was rendered unnecessary.

4. Large file sizes - Many of the files we worked with were extremely large, containing over 50,000 columns each. The 4 original Parquet files would take about 1-2 hours each to run through Pandas to extract information. There were also issues in uploading these Parquet files to our Github or sharing them among the team members. Further, after the extraction and language processing was complete, we had trouble analyzing the data in Excel because it often caused it to crash. Our FreqWords function [14] would not run on sets over 30,000 columns.

C. Solutions

1. Virginia Parquet files - To combat the issue of only having access to blog.virginia.org information, we were given new data from one of our clients, Xinyue (Cyrus) Wang. This new data was for the state of California's tourism site, visitcalifornia.com. We were initially set back from the shift in project scope, but quickly adapted and started working with the California data. With tourism being a bigger industry in California than in Virginia, came the opportunity to explore more visualization options with the given metadata.

2. Ten years of data vs. twenty - Having a smaller scope of data due to conversion issues forced us to have to explore the metadata differently. However, even with tighter bounds on the start and end date, we still had plenty of information because of the shift to the visitcalifornia.com site as opposed to virginia.org site.

3. COVID-19 Emergency - Communication and productivity became increasingly more difficult without face-to-face interactions because of the effects of the COVID-19 pandemic. However, we have utilized tools like Zoom, Google Suite, and GroupMe to continue to collaborate and keep in touch. Our group began to set smaller deadlines for ourselves to hold each other accountable and met often virtually to discuss next steps. Our clients have also been understanding of the setbacks and have worked with us to adjust our requirements accordingly.

4. Large file sizes - The majority of our solutions to the problem of having large file sizes was to simply wait them out. In the case that code would run slowly, the process would just need to be done in the background while the team member worked on other tasks. When we were unable to upload our files to our Github, we instead used Google Drive. Further, when Excel functions wouldn't run because of the size of the file, we would split up the data and run the code on smaller portions.

D. Future work

We hope our work is completed in a manner that satisfies our deliverables and also sets the project up to be taken over by future groups. The visitcalifornia.com site data is being used as an avenue to explore data extraction and visualization techniques that can be expanded in the future. We intend to pass on this information and our methods so that these same procedures can be followed on other state's tourism websites. Many of the techniques we are using and visualization ideas we are researching are universal enough to be applied to other states tourism information.

For future teams, we hope our system can provide them with tools to aid in their work. Using our data extraction and language processing techniques, more work can be done on various metadata sets for the California state tourism website. Once an adequate amount of data and visualizations are created for California, these practices can be expanded to other states and aid researchers in exploratory work on state tourism efforts.

VIII. Acknowledgements

Florian Zach, Ph.D. - Assistant Professor in the Howard Feiertag Department of Hospitality and Tourism Management
Email: florian@vt.edu

Xinyue (Cyrus) Wang - Ph.D. Student in the Department of Computer Science
Email: xw0078@vt.edu

NSF IIS-1619028, Global Event and Trend Archive Research (GETAR)

NSF CMMI-1638207, Coordinated, Behaviorally-Aware Recovery for Transportation and Power Disruptions (CBAR-tpd)

IX. References

[1] Doan, Viet, et al. “Tourism Destination Websites.” VTechWorks, Virginia Tech, 8 May 2019, http://hdl.handle.net/10919/92622, accessed 4/5/2020.

[2] “Apache Parquet.” *Apache Parquet*, Apache Software Foundation, 2018, parquet.apache.org/, accessed 4/26/2020.

[3] “Pandas - Python Data Analysis Library.” *Pandas*, NumFOCUS, 2020, pandas.pydata.org/, accessed 4/26/2020.

[4] “Natural Language Toolkit.” *Natural Language Toolkit - NLTK 3.5 Documentation*, NLTK Project, 2020, www.nltk.org/, accessed 4/26/2020.

[5] “Project Jupyter.” 2020, jupyter.org/, accessed 4/26/2020.

[6] Richardson, Leonard. “Beautiful Soup Documentation.” *Beautiful Soup Documentation - Beautiful Soup 4.9.0 Documentation*, 2020, www.crummy.com/software/BeautifulSoup/bs4/doc/, accessed 4/26/2020.

[7] “The World's Most Popular Data Science Platform.” *Anaconda*, 2020, www.anaconda.com/, accessed 4/26/2020.

[8] “Pandas.DataFrame.” *Pandas.DataFrame - Pandas 1.0.3 Documentation*, Pandas Development Team, 2014, pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html, accessed 4/26/2020.

[9] “Visual Basic Reference.” Visual Basic Reference - Microsoft VBA Documentation, documentation.help/MS-VBA-VBOB6/, accessed 4/26/2020.

[10] “Excel Help & Learning.” Office Support, Microsoft, 2020, support.office.com/en-us/excel, accessed 4/26/2020.

[11] Cronquist, Oscar. “How to Count Word Frequency in a Cell Range [UDF].” Get Digital Help, Get Digital Help, 15 Jan. 2015, www.get-digital-help.com/excel-udf-word-frequency/, accessed 4/26/2020.

[12] “SORT Function.” Office Support, Microsoft, 2020, support.office.com/en-us/article/sort-function-22f63bd0-ccc8-492f-953d-c20e8e44b86c, accessed 4/26/2020.

[13] FHTMitchell. “Trying to remove all non-English characters from list of strings”, Stack Overflow, 7 Feb. 2018,

https://stackoverflow.com/questions/48669010/trying-to-remove-all-non-english-characters-from-list-of-strings, accessed 4/26/2020.