Patent Data Analysis Report

Abhishek Jha, Boyan Tian,
Matthew Cooper, Zifan Wang

CS 4624
Multimedia, Hypertext, and Information Access
Virginia Tech
Blacksburg, VA 24061
29 April 2021
Instructor: Dr. Edward A. Fox
Client: Dr. Florian Zach

Table of Contents

Figures and Tables **2**

Abstract **3**

Introduction **4**

Requirements **5**

Design **5**

Web Scraping and Creating Database 5

Natural Language Processing 6

Implementation **6**

Web Scraping and Creating Database 6

Natural Language Processing 8

Evaluation **9**

User's Manual **10**

Developer's Manual **10**

Inventory 10

Tutorials 11

Figures 11

Methodology **14**

Lessons Learned **20**

Timeline 20

Future Work **22**

Acknowledgments **22**

References **23**

Figures and Tables

Table 1: Description of the implementation 17
Table 2: Project Timeline 20

Figure 1: Mapping data to Dictionaries 6
Figure 2: Input file example 10
Figure 3: HTML file received after GET request 11
Figure 4: Removing stop words 11
Figure 5: Removing word inflections 12
Figure 6: Removing noises 12
Figure 7: Words are vectorized 12
Figure 8: The prediction result 13
Figure 9: Input List of Assignee 15
Figure 10: Input List of Patent numbers 16
Figure 11: Methodology Overview 19
Figure 12: Tasks and deliverables 21

Abstract

The primary task was to create a database in Python, using information from either the United States Patent and Trademark Office or Google Patents, which allows efficient lookups using information such as patent assignee and the patent number. Google Patents was chosen because it contained international patent information rather than being limited to just the United States. The Jupyter Notebook was made to use Beautiful Soup to scrape data from Google Patents. The workflow of the code is to start with a user-defined comma-separated values file that specifies names, e.g., of restaurants and hotel firms, that are relevant to the analysis the user wants to conduct. The first tasks were to read in the query, create a dictionary of company names with associated patent numbers, scrape websites for lxml data, and write raw data to JSON and Excel.

The next task was to analyze the stored information qualitatively or quantitatively. Here qualitative analysis was chosen in the form of Natural Language Processing (NLP). The goal was to classify the patents using NLP. The key steps included noise removal, stop word removal, and lemmatization.

With this database, we can perform numerous types of analyses to study the effect of patents on the total valuation of companies. It is anticipated that Dr. Zach and future Computer Science students will build upon the current work and conduct additional forms of analysis.

Introduction

A patent can be defined as a government authority or license conferring a right or title for a set period, especially the sole right to exclude others from making, using, or selling an invention [1].

Patents are a means for inventors to protect their ideas and inventions. As long as a patent is valid, the patent owner — also referred to as the assignee — has complete control over its usage.

What can be patented? The United States Patent and Trademark Office (USPTO), the federal agency for granting U.S. patents, asserts in the language of the statute that any person who "invents or discovers any new and useful process, the machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent," subject to the conditions and requirements of the law [2].

It describes three types of patents: Utility patents, Design patents, and Plant patents. The Utility patents may be granted to anyone who invents, discovers, or improves any new and useful process, machine, article of manufacture, or composition of matter. Design patents may be granted to anyone who invents a new, original, and ornamental design for an article of manufacture. Plant patents may be granted to anyone who invents or discovers and asexually reproduces any distinct and new variety of plants [2].

It is intriguing to analyze data such as when the patent was filed, when it was granted, who was the filer, and what other patents were cited. This information enables us to assess the market value of companies.

The design and implementation were divided into two components: (1) web scraping and creating a database, and (2) natural language processing. Web scraping consisted of navigating to the appropriate page based on the given patent assignee or a patent number and parsing the HTML to store the pertinent information in a data structure that enables rapid lookups. On the other hand, qualitative analysis in the form of natural language processing consisted of refining the aforementioned data in order to categorize the patents.

We were particularly interested in hotel and restaurant firms and almost exclusively analyze the utility patents of such companies. The result of the project was the development of a database and analysis tools that helps with research on the effects of patents and on applying them to infer information related to tourism and businesses.

Requirements

The project deliverables included a Python script that stores in a CSV file the data, such as filing date, approval date, classification, assignee(s), inventor(s), abstract, description claim, patents cited, and figures from Google Patents.

Additionally, we were asked to conduct either qualitative analysis or quantitative analysis, described as conversion using analytical thinking to transform data to information. The former included visualization of data such as patent count, patent classification count, or concepts. The latter included analysis of the stored data with Natural Language Processing (NLP) using the NLTK toolkit to classify the patents [5].

Design

Web Scraping and Creating Database

Web Scraping is a technique employed to extract large amounts of data from websites whereby the data is extracted and saved to a local file in your computer or to a database in table (spreadsheet) format.

If we wanted to analyze the patent data or download it for use, we would not want to painstakingly copy-paste or manually write everything. Web scraping is a technique that lets us use programming to do this. It collects the data we want to work with and outputs it in the requisite format, using dictionary structures. Figure 1 illustrates the mapping of the data from the webpage into the dictionaries.

We employed a popular approach utilizing Python and the Beautiful Soup library [6]. After computing the link to navigate to, we send a GET request to the server and receive the HTML, CSS, and JS files. We isolate the tags in the HTML and get the contained text to extract all of the text inside the tag.

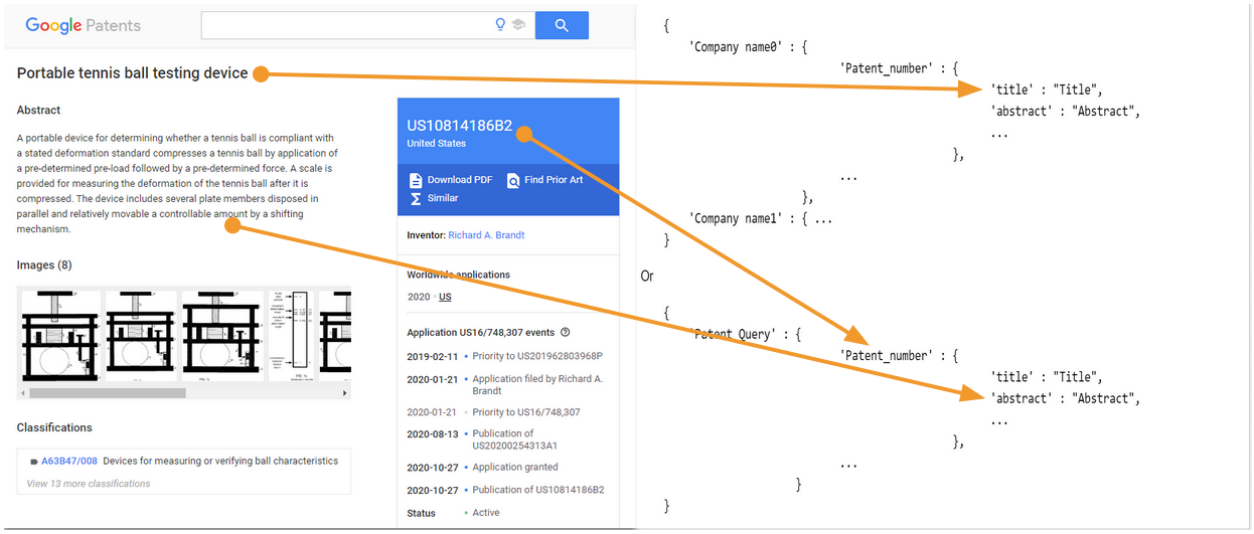

Figure 1: Mapping data to Dictionaries

Natural Language Processing

Stopword removal: Stop words are a set of commonly used words in a language. Examples of stop words in English are “a”, “the”, “is”, “are” etc. The intuition behind using stop words is to remove low information words from the text, allowing us to focus on the relevant words almost exclusively [5].

Lemmatization: The goal is to remove inflections and map a word to its root form. This transformation allows easier processing. For example, the word “better” would map to “good”. It may use a dictionary such as WordNet for mappings or some special rule-based approaches [5]. For instance, the word "better" has "good" as its lemma and the word "talk" is the base form for the word "talking".

Implementation

Web Scraping and Creating Database

Beautiful Soup is used to scrape data from Google Patents [6]. So that our solution can work for various domains, we have an input file for a given domain. Thus, for our client, we begin by

supplying a user-defined comma-separated values file that specifies names of hospitality companies that are relevant to the analyses our client wants to conduct. The script will automatically discern if the query is a list of companies or patent numbers. It uses regular expressions to match the pattern "US/d*", in order to decide. If a company name matches that expression it will confuse the script. The following steps are taken:

Step 1 is to define helper functions and metadata about the input query. This includes the date and time of the query. The helper functions handle gathering data from web pages which are identified using the unique patent number, and writing that collected data out to Excel spreadsheets.

Step 2 is to read in the user-defined CSV file with company names or patent numbers. This information is then stored in a Pandas data frame for future reference. If patent numbers are given, skip to Step 4.

Step 3 is to use the Python patent module from the patent-client library to query the USPTO full-text databases for the relevant patent numbers based on the companies given. The patient-client's patent module is an API endpoint for the USPTO full-text database advanced search function.

Step 4 is to use the Python requests library to navigate to each website. The URL for each site is built by adding patent numbers to a static URL string that Google Patents uses. When the requested page is returned to the script, the response data is sent to Beautiful Soup. It uses the helper functions to pull the desired data out of the HTML of the patent page, and builds the dictionary of information about the patent.

Step 5 is writing the raw collected data out in two formats.

1. Raw data in JSON object format that allows for easy reloading into other scripts. The schema is organized by company or patent number, descending into metadata for each patent number.
2. 8 different Excel spreadsheets that display content with a focus given in the file name. The different spreadsheets were a client request. They are organized by dates, legal events, general information, and other useful categories.

Natural Language Processing

Processing the English text involves multiple steps. Step 1 is to make all text uniform. The text is converted to lowercase alphabet. This is one of the simplest and most effective forms of text preprocessing. It is found in abundance in text mining and NLP projects; it contributes significantly to the consistency of the expected output.

Step 2 is noise removal. This means deleting characters and digits such as punctuation that are not pertinent. This is done by setting up a noise list first (the list could be [(<.*?>)]), then reading through the text data and excluding the aforementioned characters. The text is divided into individual words by splitting using the whitespace, and stored in the list. Then every string is parsed to see if it contains markup or digits. If such characters are identified, they will be removed from the string.

Step 3 is to remove stop words, which are commonly used words that could be categorized as uninformative. Examples of such stop words in English are "a", "the", "is", "are", etc. After removing those words, what is left is more directly related to the patent. To accomplish this, a stop word list is used, expanding upon the stop words from nltk.corpus [5]. The parsed word list is iterated over and the words that match an element from the stop word list are removed.

Step 4 is lemmatization. The goal is to remove word inflections and map the word to its root form so that it can be easily recognized. This is achieved through a function called Lemmatizer() that can be found in library nltk.stem (import the package called WordNetLemmatizer which belongs to nltk.stem) [5].

Step 5 is to vectorize the processed data. Thus, we need a function to transform language into numbers. Vectorizing words by using TF-IDF weighting can help us to do this by using the function called TfidfVectorizer().transform() [5]. The package called TfidfVectorizer needs to be imported from the library sklearn.feature_extraction.text. After vectorization, the data will be categorized into training and test data.

Step 6 is to send training data that has been vectorized, to the classification model. The classification model that we use is SVM. We have to first import svc from sklearn.svm, then use the function SVC(kernel='linear', probability=True).fit() to train the model [5].

Step 7 is to test the model's performance, using SVC(kernel='linear', probability=True).predict() to predict the category of patents [5] . This will output the model's predicted patent type. Comparing the actual data to the calculated data, we can evaluate the model.

Evaluation

Evaluation of this app was done in formal and informal ways. Jupyter Notebooks lend themselves well to running code and testing during writing the notebook. Testing during the implementation of this notebook was done in this form. The main issues encountered were basic errors and typos.

Following the implementation of the notebook, testing was more formal. It started with known patents that were previously analyzed for the right HTML tags containing expected information. Patents contained variable locations of the data so that the searching of the DOM was the primary function. It also established the pattern of the DOM and implementation of each Google Patent page.

Once this had been established we did testing on volume and coverage of patents. This was limited by the coverage of the Patent Client library we are utilizing [7]. That said, we tried companies that are well known. This is when we discovered a runtime issue that had to be noted. A company such as Apple has a large number of patents, which causes a significant time penalty. The cost comes in the form of pulling data from the Patent Client and then performing list comprehension to index and extract the patent numbers.

The third phase of evaluation involved scraping on edge cases and pseudo-random data to see how the app functions if a user really isn't careful to craft their query. One case considered was a company name that doesn't exist. This is managed by the code by showing an error from the Patent Client inside a try, except block. Another case is when a patent number searched for doesn't exist. This is handled at the scraping level, by displaying a 404 error from Google Patents. In both cases, execution will continue for other valid parameters, as the error is caught with a try except structure.

Another situation in the third phase of testing is when patents do not have a part of the DOM we are looking for. This is common with old patents that are missing dates, inventors, etc. The risk of patents having incorrect data was deemed low and was accepted as we moved forward. The issue may still persist but most patents have been standardized enough.

Stage four of testing was the final phase where the finished product was sent to Dr. Zach for feedback on usability and user experience. Some improvements were made as a result. Future work should continue with this, guided by Dr. Zach.

User’s Manual

The user has four jobs when using this project. The first job is to create a data file. This file is a one column CSV file with company names that are potential assignees of patents, or patent numbers. The second job is to fill in the path and filename in so that the script can reference the CSV from job one. Further instructions are provided in the Jupyter notebook. The third job is to run the notebook until the last block of code. Before running the last block of code, provide an output location for the output files to be placed. If none is specified, then the local directory will be used. Then run the final code block. That is all a client needs to do to run this Jupyter notebook. There are further instructions and information for troubleshooting in the Jupyter notebook itself.

Developer's Manual

Inventory

Data files

Input file: a one-column CSV file, similar to Figure 2 that contains companies’ names or patent numbers, such as:
US565896
US789432
…
Or
Company1
Company2
…

Figure 2 is representative of a short input file with two company names.

Figure 2: Input file example

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>US10735201B1 - Method and apparatus for key printing
        - Google Patents</title>
<meta content="width=device-width, initial-scale=1" name="viewport"/>
<meta charset="utf-8"/>
<meta content="origin-when-crossorigin" name="referrer"/>
<link href="https://patents.google.com/patent/US10735201B1/en" rel="canonical"/>
<meta content="
     System and methods for key printing may include a control panel operable to receive a mobile device identifier from a mobile device.

   " name="description"/>
<meta content="patent" name="DC.type"/>
<meta content="Method and apparatus for key printing
       " name="DC.title"/>
<meta content="2016-07-15" name="DC.date" scheme="dateSubmitted"/>
<meta content="
     System and methods for key printing may include a control panel operable to receive a mobile device identifier from a mobile device.

   " name="DC.description"/>
<meta content="US:15/212,036" name="citation_patent_application_number"/>
<meta content="https://patentimages.storage.googleapis.com/bd/41/36/39ff3bbb582558/US10735201.pdf" name="citation_pdf_url"/>
<meta content="US:10735201" name="citation_patent_number"/>
<meta content="2020-08-04" name="DC.date" scheme="issue"/>
<meta content="David M. Straitiff" name="DC.contributor" scheme="inventor"/>
<meta content="Gregory J. Durrer" name="DC.contributor" scheme="inventor"/>
<meta content="Suraj Saraf" name="DC.contributor" scheme="inventor"/>
<meta content="Neil R. Schubert, III" name="DC.contributor" scheme="inventor"/>
<meta content="Naveen Singhal" name="DC.contributor" scheme="inventor"/>
<meta content="Nathan Van Orden" name="DC.contributor" scheme="inventor"/>
<meta content="Marriott International Inc" name="DC.contributor" scheme="assignee"/>
<meta content="US:20040046018:A1" name="DC.relation" scheme="references"/>
<meta content="US:20050138387:A1" name="DC.relation" scheme="references"/>
<meta content="US:20120022902:A1" name="DC.relation" scheme="references"/>
<meta content="US:20130200999:A1" name="DC.relation" scheme="references"/>
<meta content="US:20130096963:A1" name="DC.relation" scheme="references"/>
```

Figure 3: HTML file received after GET request

Figure 3 is what the beginning of the HTML of the Google Patent page looks like. This structure, the DOM structure, is navigated and searched by BeautifulSoup to extract the information needed from the patents.

Tutorials

Before coding, install two main libraries, BeautifulSoup and nltk.

Figures

Figure 4 shows the stop word removal step. On the second line, the stop words have been replaced by a placeholder character W. On the third line, the W's have been removed.

```
['A', 'geese', 'system', 'for', 'i
['W', 'geese', 'system', 'W', 'inp
['geese', 'system', 'inputting', '
```

Figure 4: Removing stop words

Figure 5 shows removing word inflections and mapping the words to their root form.

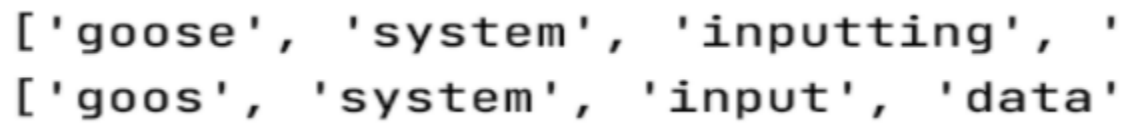

Figure 5: Removing word inflections

Figure 6 shows removing noise. The first line has the string, “specifically,” and the comma is not desired. The second line of Figure 6 shows the change to specifically without a comma. The noise can be HTML markup, non-ASCII, digits, or whitespace. Stop words are also removed at this step. Stop word removal can happen earlier, and is implementation dependent.

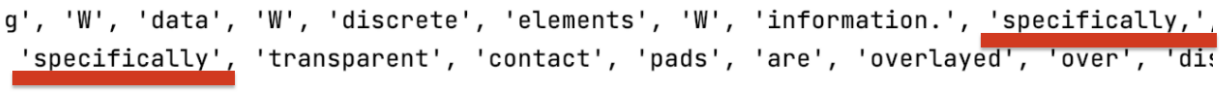

Figure 6: Removing noise

In figure 7, the first entry is the token ID that represents a word. The second entry is the TF-IDF weighting of the word in the text.

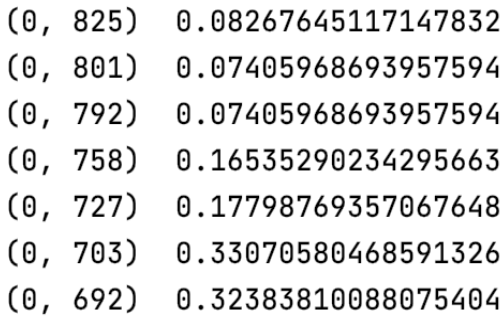

Figure 7: Words are vectorized

The first list in Figure 8 shows the patent category prediction result. The second list shows the probability of each category. For example, for the first document, the probability of being electricity is 0.95859575, and the probability of being physics is 0.0414025. So the prediction category of the first document should be electricity, which is corresponding to the result list. There are three documents in the text data, that is why we have three elements in both the result list and probability list.

```
['electricity', 'physics', 'physics']
[[0.95859575 0.04140425]
 [0.26681075 0.73318925]
 [0.16209612 0.83790388]]
```

Figure 8: The prediction result

Methodology

The nature of this system is data extraction and preparation for analysis. This allows for the system to be classified as a one dimensional system to be used by a single type of user, the researcher.

The researcher has two possible goals, as detailed below. One is to query and extract patent information from Google Patents by providing assignee names to the system. The second is to query and extract patent information from Google Patents by providing patent numbers to the system.

The extracted data is supposed to be stored in a systematic way that allows for the researcher to then move on and complete further research with the extracted data.

Goal One, Query Based On Assignee:

1. Understand User Query: Parse and store user query
    a. Read in user query: Input CSV file from user.
    b. Organize query into list: Store assignee names in list
    c. Iterate over assignee list to retrieve patent numbers: Iterate over list of patents for that assignee
2. Iteratively Make User Query: Populate dictionary
    a. Use patent number to navigate to Google Patent page: Go to patent's webpage
    b. Store patent page for data collection: scrape data
        i. Collect Data
            1. Search page for needed information using DOM hierarchy
            2. Filter out unnecessary information: Refine data, remove noise
            3. Store necessary information: Store the refined data
            4. Return to 2. Making User Query for next patent number: repeat the steps
3. Output Data
    a. Write stored information to files for future use: Create and output resultant file

The previous goal is detailed in Figure 9.

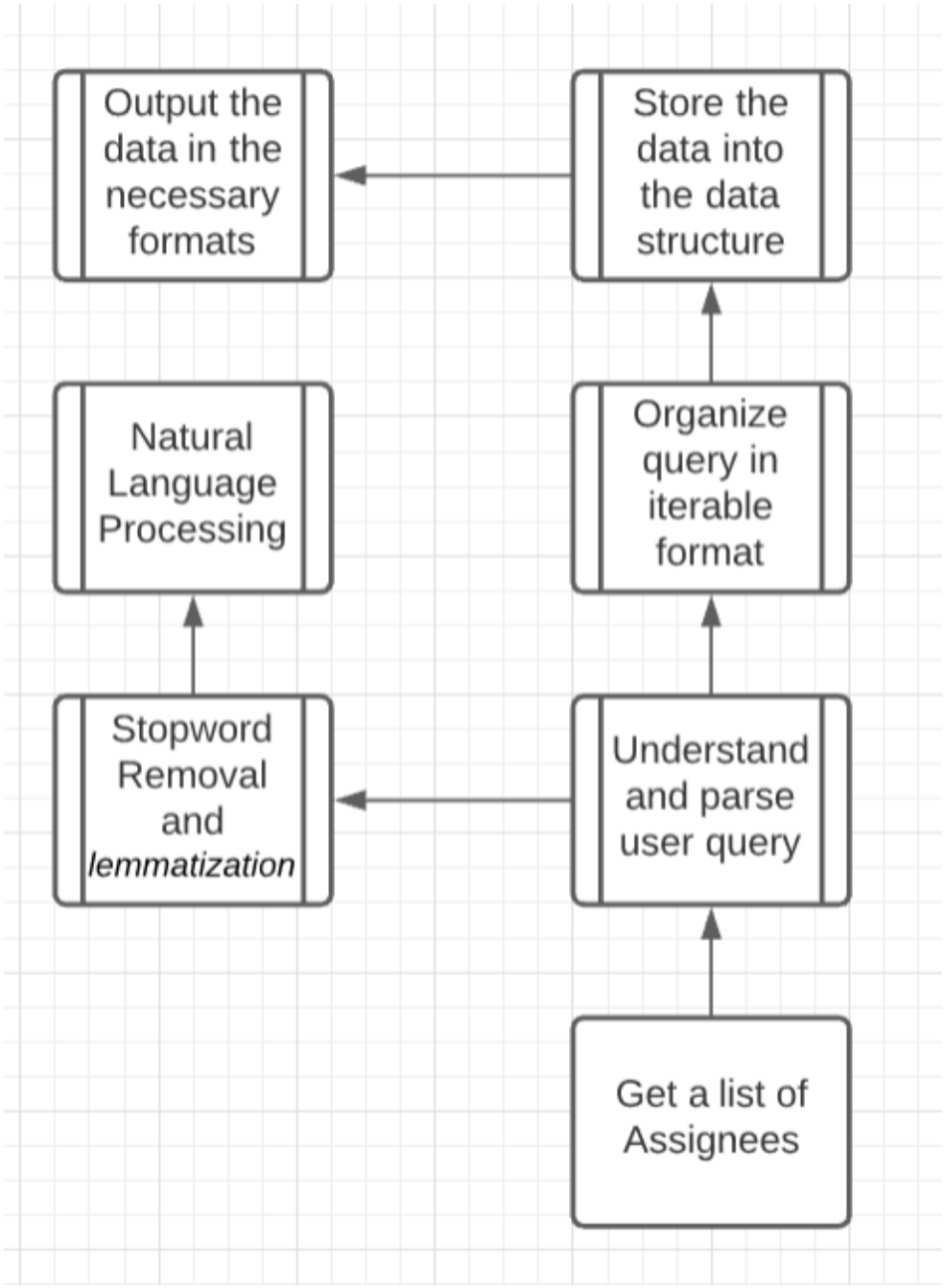

Figure 9: Input List of Assignee

Goal Two, Query Based On Patent Number:

1. Understand User Query: Input CSV file from user.
    a. Read in user query: Input CSV file from user.
    b. Organize query into list: Store patent numbers in list
2. Iteratively Make User Query: Populate dictionary
    a. Use patent number to navigate to Google Patent page: Go to patent's webpage

b. Store patent page for data collection: scrape data
    i. Collect Data
        1. Search page for needed information using DOM hierarchy
        2. Filter out unnecessary information: Refine data, remove noise
        3. Store necessary information: Store the refined data
        4. Return to 2. Making User Query for next patent number: repeat the steps

3. Output Data
    a. Write stored information to files for future use: Create and output resultant file

The previous goal is detailed in Figure 10.

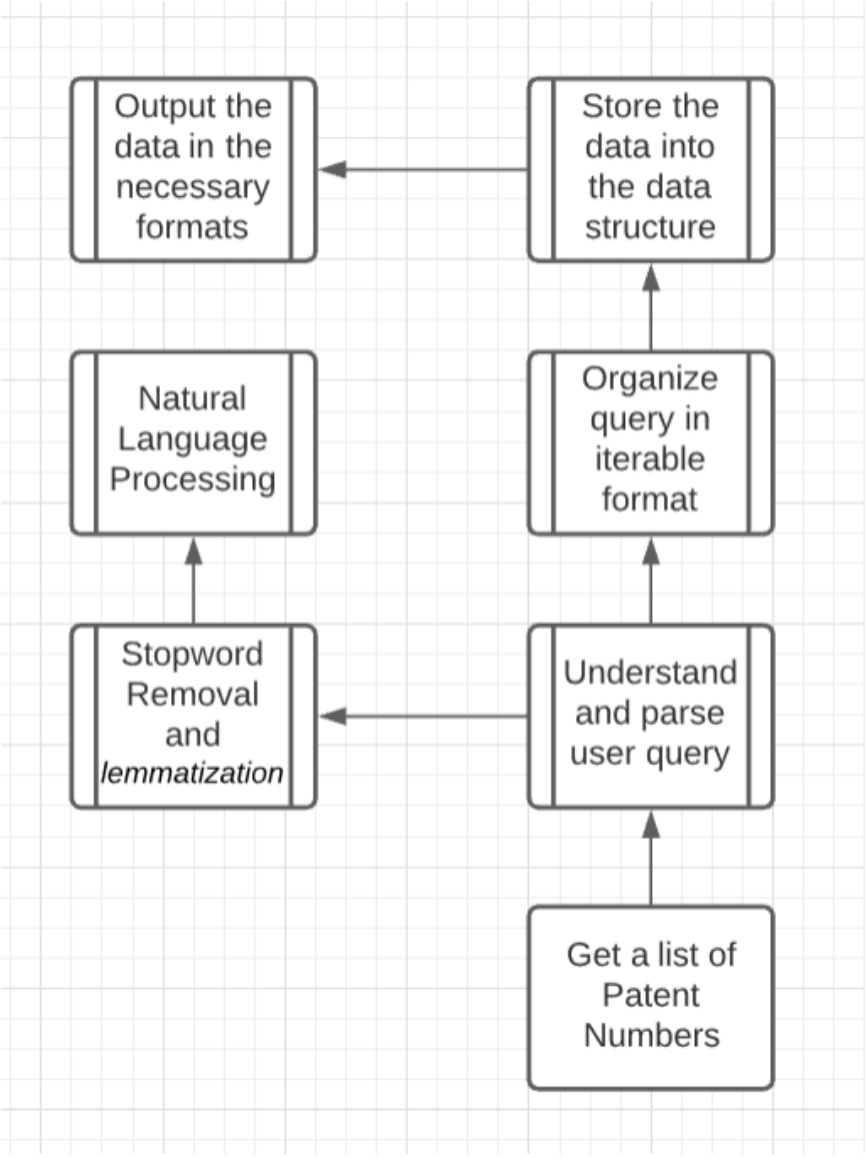

Figure 10: Input List of Patent numbers

Get a list of assignees: Parsed from the input file to navigate to the relevant webpage

Get a list of patent numbers: Parsed from the input file to navigate to the relevant webpage

Understand and parse user query: Differentiate between assignees and patent numbers so that they can be handled

Stop word Removal and lemmatization: Refine data to enable qualitative analysis

Natural Language Processing: Use the NLTK package in order to perform the qualitative analysis

Service ID	Service Name	Input file name(s)	Input file IDs (comma-sep)	Output file name	Output file ID	Libraries; Functions; Environments	API endpoint (if applicable)
S1	Service1 A	companyName.excel	CN1	Company name list	SCN1	Library: BS4 xmsl	Jupyter notebook
S2	Service1 B	patentNumber.excel	PN1	Patent number list	SPN1	Library: BS4 xmsl	Jupyter notebook
S3	Service1 C	String list (company name list or patent number list)	SCN1 OR SPN1	URL list	UL1	Library: BS4 xmsl	Jupyter notebook

S4	Service1 D	URL list	Ul1	PatentData.cvs	Pd1	Library: BS4 xmsl	Jupyter notebook
S5	Service2 A	Raw text list	RT1	Reduced Text list	SRT 1	Library: NLTK ReduceTextData	NULL
S6	Service2 B	Reduced Text list	SRT1	Noise remove list	SRT 2	Library: NLTK NoiseRemove()	NULL
S7	Service2 C	Noise remove list	SRT2	Remove-Stop-Word list	SRT 3	Library: NLTK RemoveStop-Word()	NULL
S8	Service2 D	Remove-Stop-Word list	SRT3	Finished list	SRT 4	Library: NLTK	NULL
S9	Service2 E	Finished list	SRT4	Prediction result	SRT 5	Library: NLTK	NULL

Table 1: Description of the implementation

Goal 1 (find all patents of specific companies):
Workflow1 = S1 + S3 + S4

Goal 2 (find all patents of specific patent number):
Workflow2 = S2 + S3 + S4

Goal3 (classify all specific patents):
Workflow 3 = Workflow 1 or Workflow 2 + S5 + S6 + S7 + S8 + S9 + put results into the excel file and return.
The input file is the result from workflow1 or 2. Search for the patents' abstracts and scrape from the web. Then do pre-processing for the raw data. Finally put the processed data into the classification model and the categories of each patent will be returned.

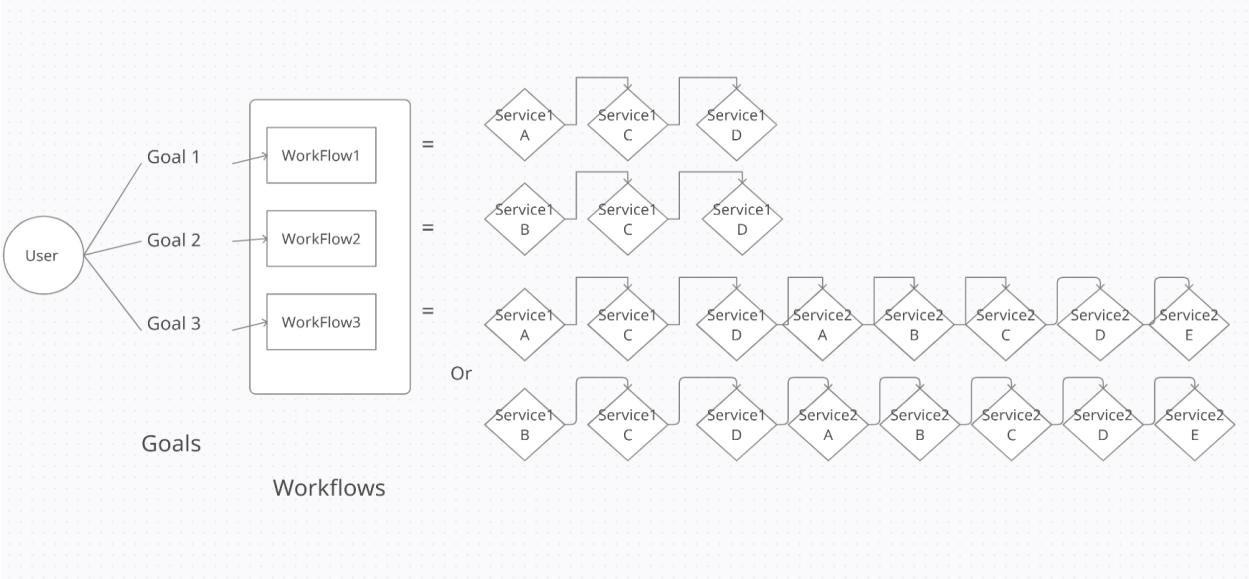

Figure 11: Methodology Overview

Figure 11 elucidates the relationship between the aforementioned Goals, Workflows and services.

Lessons Learned

Timeline

Dates:	Milestones:
01/25/2021	Study of Google API and patent databases. Potential tool list created.
02/01/2021	Second meeting and demonstration of proof of concept Python script for data demonstration.
02/02/2021	Sign off of this contract by Dr. Zach and Dr. Fox.
02/10/2021	Completion of research and understanding of tools to be used in the scripts. Database to pull US Patent data from has been decided. Start working on Python script.
02/16/2021	Presentation 1 completed and information related to the presentation shared with Dr. Zach.
03/08/2021	Progress review for testing and usage of the Python script. Work beginning on data analysis deploying, registering for VTechWorks, interim report, and presentation two.
04/01/2021	Presentation two completed. & Share related info with Dr. Zach.
04/06/2021	Methodology & Interim report Done. Final work planned and assigned for the remaining month before final report and presentation.
04/15/2021	Check-in on remaining work. Final presentation and report work is moved to top priority to ensure proper handoff of project and deliverables.
04/29/2021	The final presentation is finished and information related is shared with Dr. Zach.
05/05/2021	Final report, presentation, and project deliverables to concerned parties.

Table 2: Project Timeline

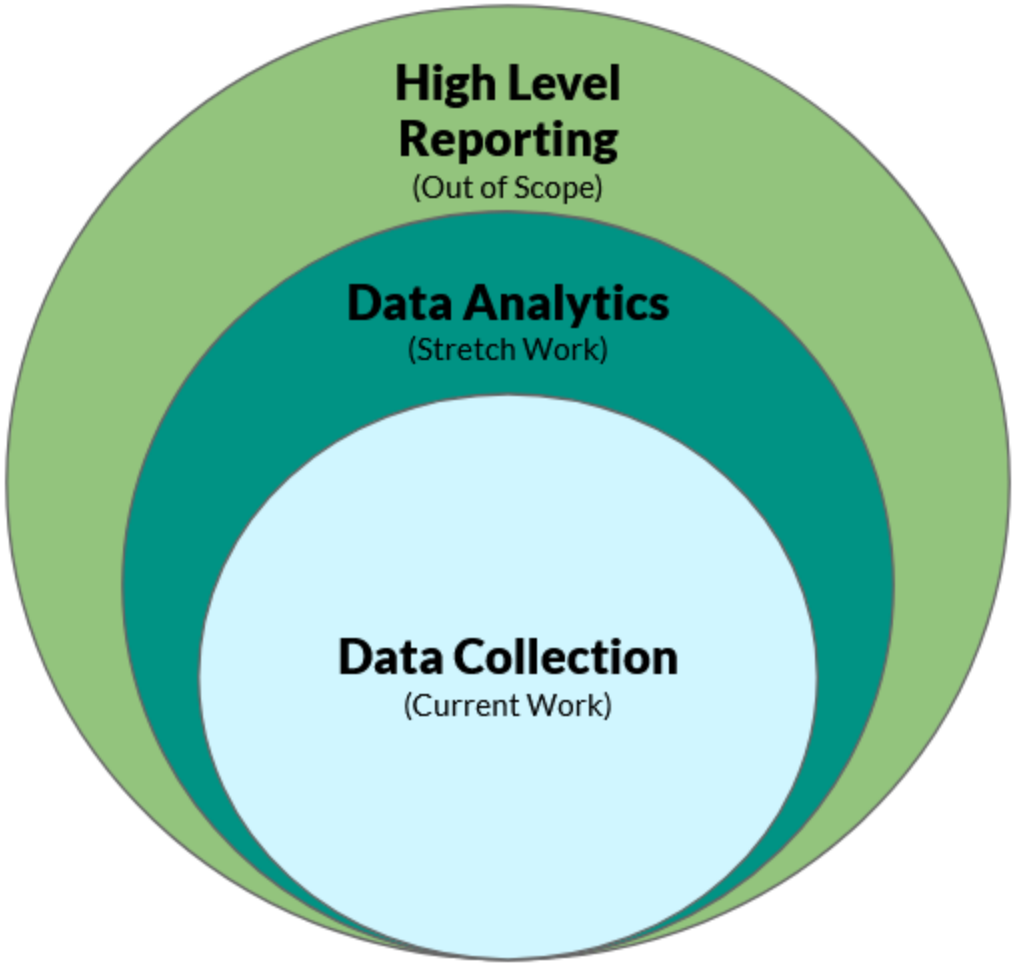

Figure 12: Tasks and deliverables

Future Work

The future work for this project will be directed by Dr. Zach. More feedback will be given to future teams according to his evaluation of the software, which was not completed this semester.

One specific future improvement is to make sure that the script is even more flexible. Early patents can have missing data that will cause issues for the script. These errors are handled, but they can cause holes in data that could be filled easier with data cleaning operations down the line, so N/A values might be appropriate.

Another improvement is the addition of more parameters and customization of the data desired from the patents. Maybe the client is only interested in the dates associated with the patents. Some way of shortening the scraping would be a challenge, but useful.

The third improvement would be to integrate some visualizations for the collected data. This can be added to the end of the Jupyter notebook, which gives the user the flexibility to run the code or not. Histograms of classifications, distance metrics, or clustering analysis could be done based on the term frequency and inverse document frequency. There is great flexibility available.

Acknowledgments

We would like to acknowledge our clients Dr. Florian Zach and Simone Blanco from the Howard Feiertag Department of Hospitality and Tourism Management along with our professor for the Multimedia, Hypertext, and Information Access class, Dr. Edward A. Fox. Additional details can be found below:

Dr. Florian Zach
Howard Feiertag Department of Hospitality and Tourism Management
florian@vt.edu

Simone Bianco
Howard Feiertag Department of Hospitality and Tourism Management
simobia@vt.edu

Dr. Edward A. Fox
Department of Computer Science
fox@vt.edu

References

[1] PATENT: Definition of patent by Oxford dictionary on LEXICO.COM also meaning of patent. (n.d.). Retrieved April 08, 2021, from https://www.lexico.com/en/definition/patent

[2] General information concerning patents. (2021, April 07). Retrieved April 08, 2021, from https://www.uspto.gov/patents/basics

[3] Zach, F., Dr. (2021, February 4). 2021Project-PatentDataAnalysis. Retrieved April 8, 2021, from https://canvas.vt.edu/courses/125164/pages/2021project-patentdataanalysis

[4] Vik Paruchuri (2021, April 05). Tutorial: Web scraping with python using beautiful soup. Retrieved April 08, 2021, from https://www.dataquest.io/blog/web-scraping-python-using-beautiful-soup/

[5] Natural language toolkit (2021, April 20). Retrieved April 08, 2021, from https://www.nltk.org/

[6] Beautiful soup documentation. (2020, October 03). Retrieved April 08, 2021, from https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[7] "Patent-Client." *PyPI*, 25 Mar. 2021, https://pypi.org/project/patent-client/