

|  |
| --- |
|  |
| Web Scrapping and Storage  Via PersonaPanels |
| |  | | --- | | Devon Conway | |

We

**Introduction**

­­­

The main objective of the final project is to determine the best ways to extract article information for a persona and store it in a non-relational database.

The specific persona that will be worked with during this project will be the ‘Younger Baby Boomers’ which is also referred to as the Generation Jones. This persona is reported to have a median age of 60 and is mainly consisting of Caucasians who primarily live in the suburbs. They typically have grown families and are known for an unhealthy work/life balance – preferring to give it all to their workplace. This persona is known to be conservative and hold religion as a central part of their moral and ethical thinking – typically self-identifying as Christians. This generation watches a large amount of television, averaging 27 hours a week and can adapt to new technologies (PersonaPanels).

The first step in the process is extracting five articles from the “Young Baby Boomers” persona and inputting them into a Microsoft Excel worksheet. In R, working with a CSV file is advantageous and can be easily loaded into the program to prepare for data cleaning. Python can also be used to clean the data – especially if the text is extracted from python. Either program will produce similar results and will be up to the data cleaner to decide. Once the data is cleaned it can be inputted into a database.

**Methods**

***Text Extraction Methods***

For the work in this report, the text was extracted by a copy-paste method and saved as a CSV file for cleaning in R. This method is not efficient and would be a time-consuming endeavor if all the articles were to be processed and entered into a database. A better method would be to extract the URLs from the personas and create a function to extract text. This can be accomplished with an API connected to the personas. Permissions would need to be granted and access permitted to the URLs for each persona.

An example code of basic text extraction in python has been provided in appendix I. The output of this code can be stored into a data frame and exported as a CSV. It is also possible to complete data cleaning in python and export the cleaned file via CSV. It is recommended that this code be part of a function or loop that can be applied to a CSV of URLs. Any URLs that do not fit the format of the loop can be added to a list that is outputted, so that the analyst can figure out what needs to be adjusted.

There are many different packages that can be used for text extraction. The one used in the example code is beautifulSoup as it offers relatively easy use.

***Data Cleaning Methods***

There are a few basic methods that need to be completed when cleaning text for processing. While there are many types of data cleaning that should be considered, these are six basic steps that need to be taken first.

The order these processes take place depends on what is being accomplished. It might be a good idea to start by removing any unwanted data, such as unnecessary punctuation. Since PersonaPanels is interested in key terms, all punctuation can be removed. It should be noted that contractions will lose their apostrophe as will hyphened words, so special care should be taken with those conditions first.

Tokenizing is breaking up strings into smaller units and can be performed next, delimited by spaces: ‘ ‘. Tokenizing is a great way to break sentences down into smaller units of words, which can make the next few steps a bit easier. It will also be easier to isolate words of interest.

Stopwards are considered words that are used so often that they tend not to mean much for sentiment analysis. Considering PersonaPanels is interested in words of interest, these are not necessary to keep. Examples of stopwards might be considered ‘the’, ‘and’, ‘for’, ‘but’, ‘a’, etc. These can be dropped from the text analysis because they will not be useful to compare in terms of finding similar interest between personas.

It is possible to correct common spelling mistakes in the text. This should take place before running any algorithms to remove words as misspelled words could be missed. Misspelled words will also not be lemmatized correctly.

Lemmatizing breaks down each word to its root or stem. An example of this could be found in the different tenses of the word ‘run’: ‘running’, ‘ran’, ‘runs’. These can all be lemmatized back to the word: ‘run’. Lemmatizing will make processes the text easier for programs.

Finally, the text can be cleared of any stopwards. Stopwards are those that are used frequently that they tend to not mean much in terms of sentiment analysis. Additionally, they will create more noise and cause inaccurate results for testing between words of interest between personas. Examples of these stopwords: ‘the’, ‘a’, ‘in’, ‘this’, etc. In some text analysis, removing specific stopwords can change the sentiment of the text; however, since keywords are of interest, any stopword can be removed which will isolate the keywords fully (Hamza, 2020).

Figure 1. An example sentence taken from one of the articles from the Young Baby Boomers Persona (Wright, 2020).

***Brief Example in R***

Data cleaning in R can be accessed through the stringr and tidyverse package. Stringr is excellent for working with strings, character manipulation, and the use of whitespace tools in data cleaning and preparation. String normalization techniques transform a variety of strings to a smaller set of string values which are more easily processed. The first step to successful clean data in R is to get rid of any unwanted space or characters within the text. At first glance of the five articles from the “Young Baby Boomers” CSV file, the titles of the articles embodied characters such as “-“ in between the words. In aiding PersonaPanels, getting rid of such characters is a must. This process can be performed to remove any unwanted character, word, number, etc. from any aspect of the article from the title to the article text.

# Remove items

> youngbabyboomers <- youngbabyboomers %>%

# Remove 'minutes' from duration: duration\_trimmed

+ mutate(title\_trimmed = str\_remove(title, "-"),

# Convert duration\_trimmed to character: duration\_mins

+ title\_trimmed = as.character(title\_trimmed))

Figure 2. Example code to remove unwanted characters in R.

The example code in figure 2 can be used to remove unwanted data in the article texts. Additional packages may be used to complete the rest of the steps. Moreover, the tidyr package in R allows for more control over a specific dataset and goal that one if trying to achieve when cleaning data before inputting the data into a database. If PersonaPanels was looking to merge the article ID and article title, the unite() function in R within the tidyr package allows for uniting numerous columns into one by pasting the strings together. Finally, the nesting technique could be utilized within the database for PersonaPanels to make it simple and clear if that is their vision. Nesting allows for a representation of grouped data where a group, such as titles, can be nested into a single row containing a data frame of titles.

While cleaning data for text analysis is possible in R, typically python is preferred. For python, there are a couple packages available that are preferred for natural language processing (NLP). Natural Language Toolkit (NLTK) is a well-known package because it provides a wide variety of functions such as: sentence detection, tokenization, lemmatization, stemming, parsing, chunking, and part-of-speech tagging (Goyal, 2020). Another great package is TextBlob which offers spelling corrections as well as everything else NLTK offers, additionally, TextBlob is a great tool for beginners in NLP (Jain, 2018).

***Database Recommendations***

A search engine database is optimal for Persona Panels in that it allows applications to search for information and key worlds that are held in external data sources. The articles filtering through the “Younger Baby Boomers’ persona are changing daily, therefore the search engine database can index large quantities of data and provide live, real-time access. According to Microsoft, indexes can be multi-dimensional and can support free-text searches across large volumes of text data. Specifically, the search engine database uses two models: the pull model and the push model. The pull model can allow for indexing fueled by the database while the push model is fueled by an external code. That being said, a Azure Cognitive Search, an AI powered cloud search service for mobile and web app development that is free to use will be the database used to hold the specific five articles from “Young Baby Boomers”. Specifically, the Azure Cosmos DB is a NoSQL database that allows for easy and open-source API’s from all over the world. This would be the specific database recommendation for PersonaPanels in that it guarantees a single digit millisecond response and a 99.9 percent availability with fast writes and instant scalability. The Azure Cosmos database is not free, but this simple ERD can showcase the proper steps of putting article text into a database. ERD diagrams are typically not used in non-SQL databases, however, they can be used as a framework to map a database.

Diagram

Description automatically generated

Figure 3. An ERD diagram of a recommended database structure for storage of text.

In figure 3, the simple ERD can aid in Persona Panels input the articles into the database. The challenges foreseen involve inputting code to clean all the articles that are in the database. This can be time consuming and costly for a company. It would be a lot easier in R if one was able to use a single function to specifically call on a certain column and automatically proceed through the data cleaning methods. Additionally, Persona Panels will need to pay to have access to Azure Cognitive Search. This can possibly be a barrier if the cost adds up over time.

**Conclusion**

Finding ways to efficiently extract texts are important to this project. The recommended method would be to combine the example python code with a function or loop and then apply it to a CSV of saved URLs. URL extraction might be difficulty as the method used in this project was copy and paste from the hyperlink provided with the persona profiles. However, if the URL was easier to expose or if Tanjo was able to update their design to allow API access to this information, then extraction could be greatly expediated.

The articles will need to go through a strict cleaning process which leaves behind only those keywords to be used in comparison across the personas. Recommended packages to use include the NLTK or TextBlob. An important note that was not considered in this paper but would need to be addressed depending on how personas analyze article information, is how to analyze any photographs or digital media that come with the articles. Typically, neural network algorithms are used to analyze complex data, but this might not be necessary if the personas ignore any non-text data.

In terms of database selection, a search engine type database would be best for the company to use as it allows access to data stored externally. Microsoft Azure is the recommended platform to store this data. Microsoft provides excellent customer support as well as high security. PersonaPanels could create their own database, but this might prove to be somewhat more costly than using Microsoft Azure. Possible issues that come with creating databases: the company is solely responsible for data storage, if there is a failure and the data is not adequately backed up, the data is gone; it will take time and money to design the database – it might need to be outsourced to a company that handles database design; PersonaPanels will need to invest in both physical storage and maintenance in order to keep the database up to date.

References

Beautiful Soup Documentation¶. (2020). Retrieved October 19, 2020, from <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

D.C. (2020). *Cleaning Data with R*. DataCamp. <https://www.datacamp.com/users/sign_in?redirect=https://learn.datacamp.com/skill-tracks/importing-cleaning-data-with-r>

Extract text from a webpage using BeautifulSoup and Python. (2019, February 12). Retrieved October 19, 2020, from <https://matix.io/extract-text-from-webpage-using-beautifulsoup-and-python/>

Goyal, K. (2020, January 01). Top 7 Python NLP Libraries. Retrieved October 21, 2020, from <https://www.upgrad.com/blog/python-nlp-libraries-and-applications/>

Hamza, A. (2020, September 30). Effectively Pre-processing the Text Data Part 1: Text Cleaning. Retrieved October 21, 2020, from <https://towardsdatascience.com/effectively-pre-processing-the-text-data-part-1-text-cleaning-9ecae119cb3e>

Jain, S. (2018, February 11). NLP For Beginners: Text Classification Using TextBlob. Retrieved October 21, 2020, from <https://www.analyticsvidhya.com/blog/2018/02/natural-language-processing-for-beginners-using-textblob/>

PersonaPanels. (n.d.). Younger Baby Boomers. Retrieved October 11, 2020, from <https://ai.delvinia.com/personas/rtNh>

Wright, J. (2020, October 10). Coronavirus UK: Britain's envoy to WHO tells government to stop locking-down. Retrieved October 21, 2020, from <https://www.dailymail.co.uk/news/article-8825949/Coronavirs-UK-Britains-envoy-tells-government-stop-locking-down.html>

Appendix I

# Installing packages

!pip install beautifulsoup4

import pandas as pd

!pip install lxml

import requests

# Importing url csv file with first link.

url = pd.read\_csv('url.csv')

url.head()

url1 = url.iloc[0,0]

url\_res = requests.get(url1)

html\_page = url\_res.content

url1\_soup = BeautifulSoup(html\_page, 'html.parser')

text = url1\_soup.find\_all(text = True)

set([t.parent.name for t in text])

output = ''

blacklist = [

'[document]',

'a',

'b',

'body',

'div',

'fieldset',

'font',

'form',

'h1',

'h2',

'h3',

'head',

'label',

'legend',

'li',

'link',

'meta',

'noscript',

#'p',

'script',

'span',

'strong',

'style',

'table',

'td',

#'time',

#'title',

'tr',

'ul',

'video'

]

for t in text:

if t.parent.name not in blacklist:

output += '{}'.format(t)

print(output)