Scraping, Storing, Visualizing and Exploring Data: Preparing Public Health Apps for Text Classification

Indicate Submission Type: Short Paper

Timoteo Kelly, Dr. Dario Bonaretti

Nova Southeastern University

Abstract

This research paper presents a process for collecting, storing, processing, and analyzing review data from over 100 public health apps on the Google Play Store to facilitate the Theory of Effective Use in the design of public health apps through the application of text classification and machine learning to the collected reviews. The paper describes the process employed, including the use of a web scraper, Google Play Scraper API, OpenPyxl, MongoDB, and SQLite3 databases. The benefits and limitations of each database are discussed. Exploratory data analysis was conducted, including the creation of a histogram to display the distribution of review scores, the use of ggplot2 and tidyverse to isolate reviews based on their word count, the use of Amelia to create a visual representation of missing or NA rows, and the use of tidytext to search for the most commonly used words in the review data. The top 20 most frequently used words were plotted using ggplot2.

Introduction

The primary objective of this study is to present a comprehensive account of the steps taken to collect, store, process, and analyze review data from Google Play Store for over one hundred Public Health apps. The purpose of this task is to facilitate the Theory of Effective Use in the design of Public Health apps through the application of text classification and machine learning to the collected reviews. To achieve this goal, we employed a web scraper using Google Play Scraper API and OpenPyxl packages in Python. We utilized MongoDB and SQLite3 databases to store the collected data for analysis. In this paper, we describe the process employed and discuss the benefits and limitations of each database. Additionally, we utilized in-memory databases to create temporary databases to store data efficiently.

Method

This study describes a comprehensive process for data collection and storage of reviews from the Google Play Store, pertaining to 146 Public Health applications. Visual Studio Code was utilized as the primary working environment, while JupyterNotebook with Python was used for the initial data collection. Surface data from Public Health applications was recorded in an Excel spreadsheet, containing basic information such as application title, URL, application date, etc.

The object-oriented programming language, Python, was used to build a web scraper using Google Play Scraper API and OpenPyxl packages. The reviews were stored in both MongoDB and SQLite3, which are object-oriented and relational databases, respectively. Other Python packages used were Datetime, Pprint, and Pandas for manipulating dates and time, printing data structures, and data structure and manipulation, respectively.

The identified Public Health applications were loaded into a Pandas data frame for easy viewing and to ensure that all data was entered correctly. At this stage, it was important to verify that the applications added to the list actually contain reviews or are still functional, otherwise, it would cause the loop to stop.

The next step involved setting up the database to store the reviews. Two options were experimented with, namely, MongoDB and SQLite3. MongoDB is an object-oriented database service that can be easily interacted with and queried using a NoSQL-based language. It stores data as documents similar to a JSON file and provides easy data sharing, data analysis, and data visualization options within its web-based

interface. On the other hand, SQLite3 is a relational database package embedded in the Python library, which provides a lightweight disk-based database that doesn't require a separate server process.
Both databases were evaluated based on their storage capacity, functionality, and ease of use. MongoDB can store up to 950,000 documents before hitting the storage threshold, whereas SQLite3 can store up to 281 terabytes of data, limited only by the system. An in-memory database was created using SQLite3 for this project, as it was perfect for creating temporary databases that are only stored in RAM, making storing and querying data faster.
Once all the Public Health mobile application reviews were scraped and stored in the temporary SQL database, the data was loaded into a Pandas data frame for data cleaning such as removing duplicates and empty rows. The cleaned data was exported to both CSV and JSON file formats, as the CSV format was quick and easy to view, but couldn't contain all of the data, while the JSON file format could contain all of the data.

Exploratory Data Analysis

The process of conducting exploratory data analysis involved importing the json file containing review data into RStudio, followed by using the gglpot2 package from the tidyverse to create a histogram of the distribution of review scores.
The histogram in [Figure 1] reveals that the highest score is 5, which is also the most frequently assigned score with 906,288 reviews. The next highest score is 4 with 236,308 reviews. The lowest score is 1 with 143,936 reviews. Score 3 has 75,874 reviews and score 2 has 59,963 reviews.
The next stage involved using the ggplot2 and tidyverse packages to isolate reviews based on their word count. This required counting each word in every review and creating a new column to display the word count. The resulting figure displays the distribution of review scores for all reviews over a 50-word count.
[Figure 2] indicates that longer reviews are more likely to be negative, as the largest group of reviews with a count of 21,537 were assigned a score of 1, which is more than double the next largest group.
The Amelia package was then utilized to create a visual representation of any missing or NA rows in the dataset. The resulting [Figure 3] highlights the repliedAt, replyContent, and CreatedVersion columns as having missing data. This is to be expected, as not every review will have a reply, and the repliedAt column contains the date for the reply, while the replyContent column contains the text. Additionally, CreatedVersion is not always available for every application in the Google Play store.
The next package used was tidytext to search for the most commonly used words in the review data. The top 20 most frequently used words were then plotted using ggplot2 and shown in [Figure 4].
A dictionary for the term "time" was then created, which included words such as "delay," "update," and "limit." Reviews containing these parameters were then extracted. The resulting figure shows that the distribution of reviews based on the defined parameters of time skews towards a score of 1 or negative (see [Figure 5])..
The genderdata package was utilized to estimate the gender of review users based on their names. This package uses data collected from Social Security from 1930 to 2012 to determine the gender of a name given. Since the review data contains usernames from Google accounts, which typically use an individual's real name (but often do not), it was a reasonable set of data to apply this tool to. The figure (see [Figure 6]) indicates that more males leave reviews on Public Health mobile applications than females.
Lastly, multiple regression analysis was performed using the add-on SigmaXL package in Microsoft Excel to examine possible correlations between review scores and word count. The results (see [Figure 7]) show a low R-Square of 0.04%, indicating a weak correlation between word count and score. Although the analysis was performed on reviews over 50-word count, if lower counts had been included in the sample data, the correlation might have been higher.

Figures

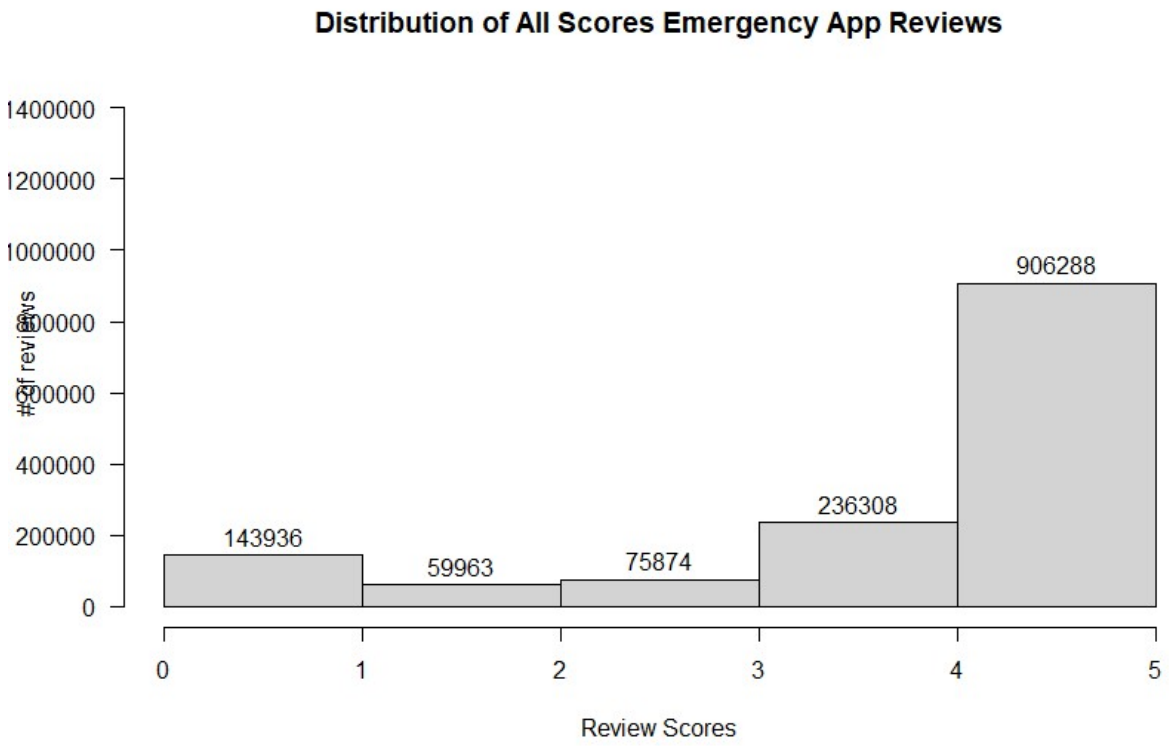

Figure 1

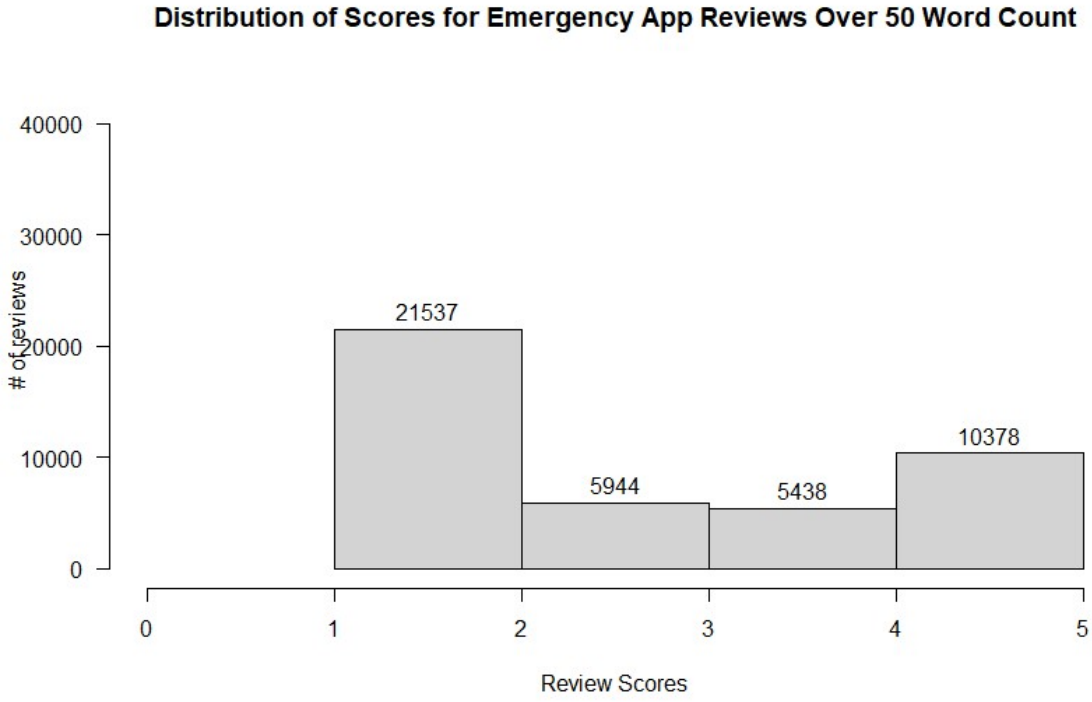

Figure 2

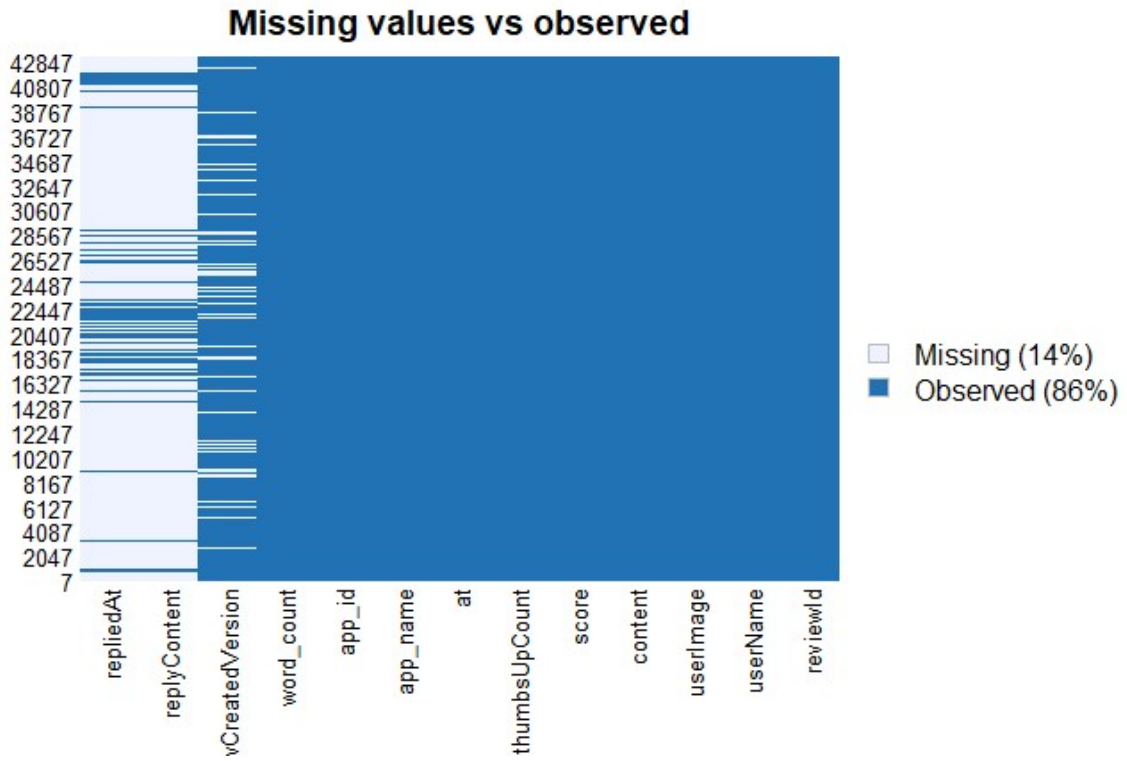

Figure 3

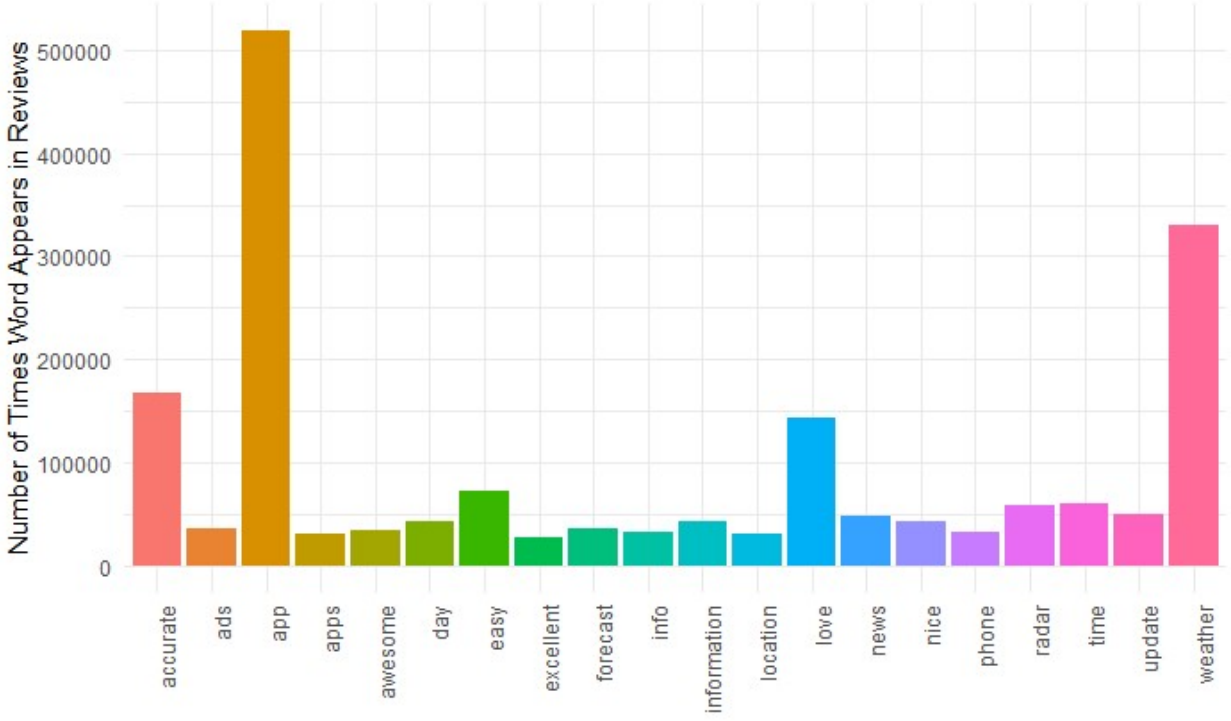

Figure 4

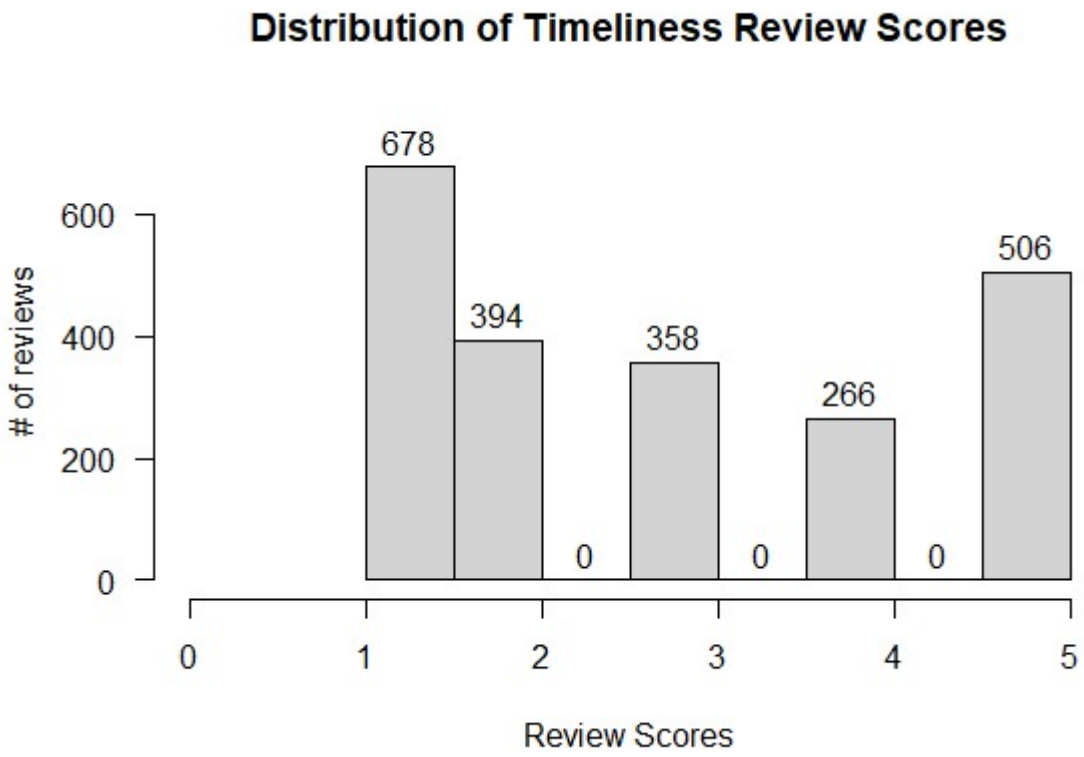

Figure 5

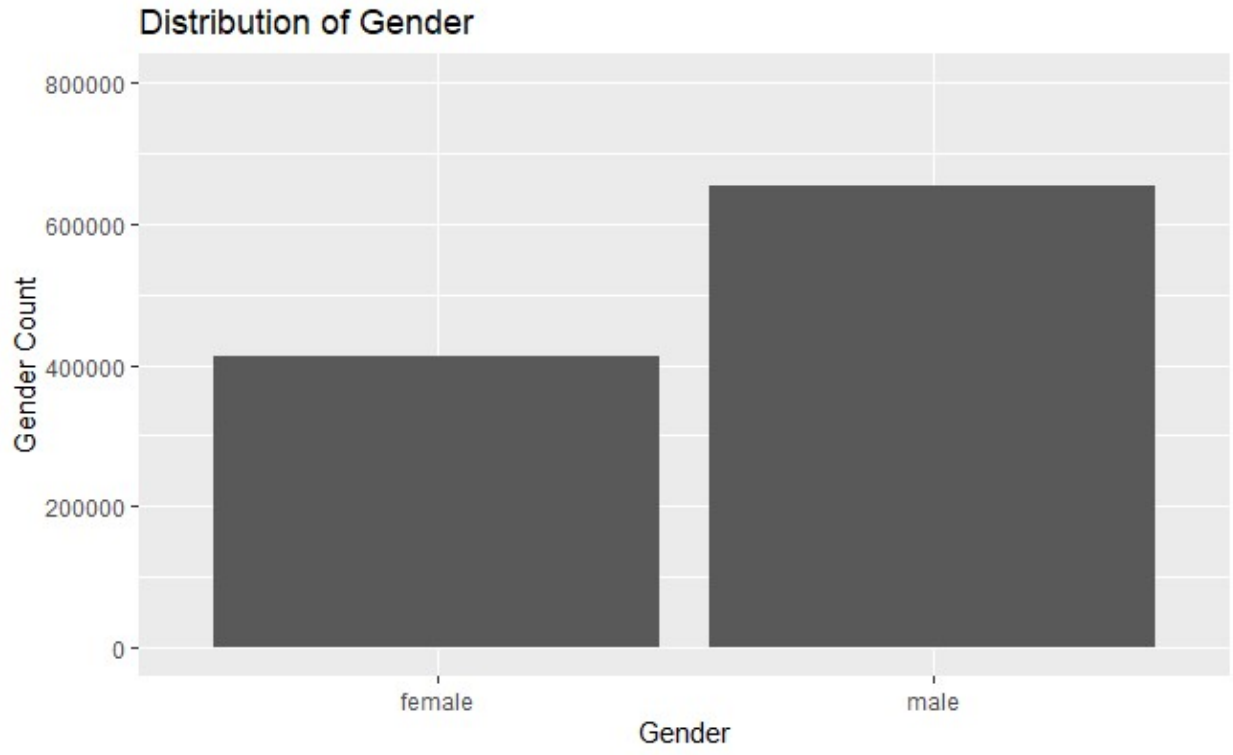

Figure 6

Multiple Regression Model: score = (2.607) + (-0.00149) * Word.Count

Model Summary:	
R-Square	0.04%
R-Square Adjusted	0.00%
S (Root Mean Square Error)	1.566

Parameter Estimates:

Predictor Term	Coefficient	SE Coefficient	T	P	VIF	Tolerance
Constant	2.607	0.125926	20.703	0.0000		
Word.Count	-0.001485466	0.002456997	-0.604586	0.5456	1	1

Analysis of Variance for Model:

Source	DF	SS	MS	F	P
Model	1	0.896500	0.896500	0.365524	0.5456
Error	998	2447.7	2.453		
Lack of Fit	78	176.28	2.260	0.915362	0.6830
Pure Error	920	2271.5	2.469		
Total (Model + Error)	999	2448.6	2.451		

Durbin-Watson Test for Autocorrelation in Residuals:

DW Statistic	1.958
P-Value Positive Autocorrelation	0.2522
P-Value Negative Autocorrelation	0.7478

Figure 7

```
from monkeylearn import MonkeyLearn

ml = MonkeyLearn('1e733b6a2d02ef8f16b73a4dcefe7d6beb5d38ff')
data = ["it could be set up with a little bit more user friendly inter
model_id = 'cl_7cSmAWxP'
result = ml.classifiers.classify(model_id, data)
print(result.body)
```

✓ 0.3s Python

```
[{'text': 'it could be set up with a little bit more user friendly
interface and start with the current/ future radar instead of being on
default past radar bit im happy with it', 'external_id': None, 'error':
False, 'classifications': [{'tag_name': 'Currency', 'tag_id': 124243233,
'confidence': 0.812}]}]
```

Figure 8

Continued Research

Despite completing the exploratory data analysis, the text classification based on the parameters of Trustworthiness, Timeliness, and Situational Awareness remains incomplete and beyond the scope of this project. However, data annotation based on the aforementioned dimensions has nearly concluded through the employment of the software as a service, MonkeyLearn. The current predictive model can proficiently label reviews according to Trustworthiness, Timeliness, and Situational Awareness with an accuracy of up

to 87%. Additional annotation of data is required to improve the classification model's accuracy to above 95%.

Displayed in [Figure 8] is a sample from our review data that underwent processing via our text classification model in Visual Studio Code. The predictive model accurately determined that the review was discussing "the present state of the real world and its representative data structures" (Bonaretti & Fischer-Preßler, 2021, p. 8), which corresponds to the definition of Currency established in Timeliness, Trustworthiness, and Situational Awareness: Three Design Goals for Warning with Emergency Apps. The level of accuracy for this prediction reached 81.2%. With further data annotation, we anticipate that results will surpass 95%.

Researchers who lack the time or expertise to annotate their training data or construct and train a model have found MonkeyLearn to be a valuable tool. The software seamlessly integrates the process into a no or low code format while remaining versatile enough to be employed in conjunction with a programming language, as demonstrated above in Python. One individual can annotate, construct, train, execute their model, and obtain results within one to two days, a task that would have previously required at least a week for a single person.

Conclusion

The methodology outlined in this study for data scraping, storage, transformation, and analysis presents a highly efficient and streamlined approach for researchers and analysts seeking to collect data from multiple applications available on the Google Play store. Moreover, the emergence of software as a service (SaaS) has provided cost-effective solutions to smaller research teams lacking resources to perform data annotation or entry tasks. The meticulously designed scripts executed in either R markdown file in RStudio or Jupyter notebooks in Visual Studio Code can be readily accessed in our GitHub repository, which is provided as a reference below.

References

"Tidyverse packages." https://www.tidyverse.org/packages/ (accessed Mar. 28, 2023).

"Text Analytics," *MonkeyLearn*. https://monkeylearn.com/ (accessed Mar. 28, 2023).

"sqlite3 — DB-API 2.0 interface for SQLite databases," *Python documentation*. https://docs.python.org/3/library/sqlite3.html (accessed Mar. 28, 2023).

"A Simpler Way to Find Your Files." https://here.r-lib.org/ (accessed Mar. 28, 2023).

tkelly1107, "Effective Use Mobile Apps for Public Health Research Project." Nov. 03, 2022. Accessed: Mar. 28, 2023. [Online]. Available: https://github.com/tkelly1107/Emergency-Warning-App

G. D. Queiroz *et al.*, "tidytext: Text Mining using 'dplyr', 'ggplot2', and Other Tidy Tools." Jan. 07, 2023. Accessed: Mar. 28, 2023. [Online]. Available: https://CRAN.R-project.org/package=tidytext

F. Olano, "google-play-scraper." Mar. 27, 2023. Accessed: Mar. 28, 2023. [Online]. Available: https://github.com/facundoolano/google-play-scraper

L. Mullen, "gender." Mar. 13, 2023. Accessed: Mar. 28, 2023. [Online]. Available: https://github.com/lmullen/gender

J. Honaker, G. King, and M. Blackwell, "Amelia II: A Program for Missing Data," *Journal of Statistical Software*, vol. 45, pp. 1–47, Dec. 2011, doi: 10.18637/jss.v045.i07.

E. Gazoni and C. Clark, "Openpyxl-a Python Library to Read," *Write Excel*, 2010.

D. Bonaretti and D. Fischer-Preßler, "Timeliness, Trustworthiness, and Situational Awareness: Three Design Goals for Warning with Emergency Apps," 2021.