International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

A Study of Sentiment Analysis on Mobile Applications

Lakshmi Srinivas Rao [a] ***, Sravan Kumar*** [b]***, Nagender Reddy*** [c]***, Jayavardhan Reddy*** [d]***, Buvanne*** [e]

[a] *Associate Professor, Anurag University, Hyderabad, India*
[b,c,d,,e] *Student, Anurag University, Hyderabad, India*

DOI: https://doi.org/10.55248/gengpi.5.0524.1325

A B S T R A C T

— In the contemporary digital landscape, user feedback on platforms like the Google Play Store holds immense value for developers and consumers alike. However, manually analyzing the sentiment of a large volume of reviews can be time-consuming and tedious. To address this challenge, we present an automated approach leveraging web scraping with RSelenium and sentiment analysis with the AFINN lexicon to analyze reviews of a specified app on the Google Play Store. Our methodology involves using RSelenium, a web scraping tool, to navigate to the target app's page, extract reviews, and then perform sentiment analysis on the collected text data. We then employ the AFINN lexicon, which assigns sentiment scores to individual words, to quantify the sentiment expressed in each review. Positive and negative sentiments are categorized based on the polarity of the sentiment scores. The implementation involves several steps, including establishing a connection to a web browser using RSelenium, locating and extracting reviews from the Google Play Store page, cleaning and tokenizing the text data, and finally, conducting sentiment analysis and visualizing the results. The project's outcomes include insights into the overall sentiment of the reviews, highlighting aspects that users appreciate or find lacking in the app. These insights can inform developers' decisions regarding app improvements and feature enhancements. Additionally, the automation of sentiment analysis reduces the manual effort required, enabling efficient monitoring of user feedback, and facilitating data-driven decision-making processes.

Keywords: Sentiment Classification, Web Scraping, AFINN Lexicon, Natural Language Processing, Data Visualization.

INTRODUCTION

In the contemporary digital era, mobile applications (apps) have become ubiquitous tools for a wide range of activities, spanning from entertainment to productivity. With millions of apps available for download on platforms like the Google Play Store, the landscape is highly competitive, with developers constantly striving to improve user experience and satisfaction. User feedback, in the form of reviews and ratings, plays a pivotal role in shaping the success and reputation of these apps. Analyzing the sentiment expressed in these reviews provides valuable insights into user preferences, satisfaction levels, and areas for improvement.

However, manually analyzing a large volume of reviews can be a daunting and time-consuming task. To address this challenge, automated techniques have emerged as a promising solution, leveraging web scraping and natural language processing (NLP) technologies. In this paper, we present a novel approach for automating sentiment analysis of Google Play Store reviews using RSelenium, a web scraping tool, in conjunction with the AFINN lexicon, a sentiment analysis resource. The primary objective of this research is to demonstrate the effectiveness and efficiency of our automated sentiment analysis approach in extracting meaningful insights from Google Play Store reviews. By automating the process, we aim to streamline the analysis workflow, enabling developers and stakeholders to efficiently monitor user feedback, identify trends, and make informed decisions regarding app improvements and feature enhancements.

This paper is structured as follows: first, we provide an overview of related work in the fields of web scraping, sentiment analysis, and automated text analysis. Next, we describe the methodology employed in our approach, detailing the implementation steps and key technologies utilized. We then present the results of our sentiment analysis, showcasing the insights gained from analyzing a sample dataset of Google Play Store reviews. Through this research,

we aim to contribute to the growing body of literature on automated sentiment analysis and its applications in the context of app development and user feedback analysis. By harnessing the power of automation and NLP technologies, we believe our approach holds significant potential for enhancing the efficiency and effectiveness of sentiment analysis workflows, ultimately leading to improved app quality and user satisfaction.

II. RELATED WORK

In this section, we introduce the most related work from five perspectives: 1) Automated Sentiment Analysis 2) Web Scraping and Text Mining 3) Sentiment Lexicons 4) Application in App Reviews 5) Automated Decision Support.

A. Automated Sentiment Analysis Prior research has explored various approaches to automate sentiment analysis of text data from diverse sources, including social media platforms, product reviews, and online forums. Techniques range from lexicon-based methods to machine learning models trained on labeled datasets. Wang et al. (2012) proposed a sentiment analysis framework for microblogging platforms, leveraging machine learning algorithms to classify tweets into positive, negative, or neutral sentiments. Liu et al. (2015) introduced a hybrid approach combining lexicon-based sentiment analysis with machine learning techniques for sentiment classification of online product reviews.

B. Web Scraping and Text Mining Web scraping techniques have been extensively used for data collection from online sources, facilitating text mining and analysis of unstructured textual data. Libraries such as BeautifulSoup in Python and RSelenium in R provide tools for automated web scraping. Gencoglu et al. (2017) employed web scraping to collect data from TripAdvisor for sentiment analysis of hotel reviews, demonstrating the effectiveness of automated data collection in sentiment analysis tasks.

C. Sentiment Lexicons Sentiment lexicons, such as AFINN, SentiWordNet, and VADER, provide pre-defined sentiment scores for words, enabling sentiment analysis without the need for labeled training data. Hu and Liu (2004) introduced SentiWordNet, a lexical resource for opinion mining, which assigns sentiment scores to synsets in WordNet based on their positivity, negativity, and neutrality. Hutto and Gilbert (2014) developed VADER (Valence Aware Dictionary and Sentiment Reasoner), a lexicon and rule-based sentiment analysis tool specifically designed for social media text.

D. Application in App Reviews Studies have explored sentiment analysis of app reviews to understand user sentiments, identify common issues, and prioritize feature enhancements. Kastner et al. (2013) conducted sentiment analysis of app reviews on the Apple App Store, highlighting the importance of user feedback in app development and quality assurance. Xiong et al. (2016) utilized sentiment analysis techniques to analyze user opinions and sentiments expressed in app reviews, providing insights for app developers to improve user satisfaction and engagement.

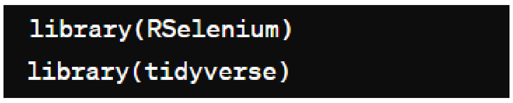
E. Automated Decision Support

Automated sentiment analysis has been integrated into decision support systems to aid stakeholders in making informed decisions based on user feedback and sentiment trends. Zhang et al. (2018) proposed a decision support system for product feature prioritization, integrating sentiment analysis of customer reviews to identify key features driving customer satisfaction and dissatisfaction. Kim et al. (2020) developed a decision support framework for app developers, incorporating sentiment analysis of app reviews to prioritize bug fixes, feature enhancements, and user support requests.

III. METHOD

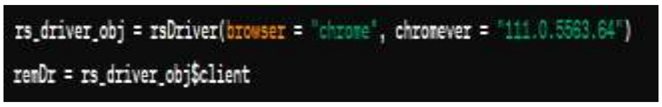
A. Web Scrapping and Data Collection By utilizing the RSelenium package in R to automate web browsing and extract data from the Google Play Store. Specifically, we targeted the reviews section of the "Racing Street" application (ID: com.aim.racingstreet) for analysis. The following steps were undertaken:

1. Setting up RSelenium Environment: Loading the Rselenium library and tidyverse library for data manipulation.



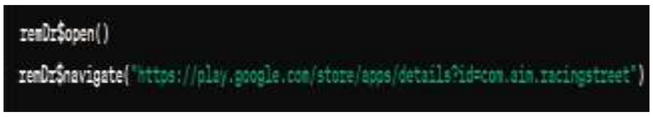
```
library(RSelenium)
library(tidyverse)
```

2. Creating RSelenium Driver Object: Instantiating an RSelenium driver object specifying the Chrome browser and its version.



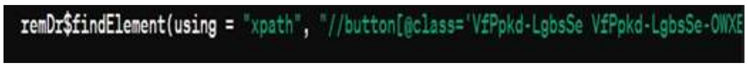
```
rs_driver_obj = rsDriver(browser = "chrome", chromever = "111.0.5563.64")
remDr = rs_driver_obj$client
```

3. Opening Web Browser and Navigating: Opening a new web browser instance and navigated to the Google Play Store URL for the target application.



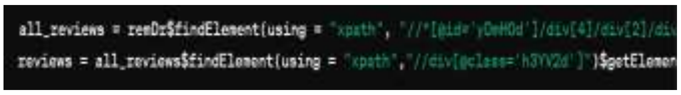
```
remDr$open()
remDr$navigate("https://play.google.com/store/apps/details?id=com.aim.racingstreet")
```

4. Interacting with Web Elements: Utilizing XPath, we located and interacted with specific elements on the webpage. This included clicking a button to expand all reviews.



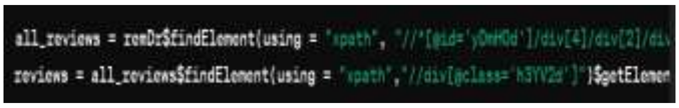
```
remDr$findElement(using = "xpath", "//button[@class='VfPpkd-LgbsSe VfPpkd-LgbsSe-OWXE
```

5. Scraping Reviews: Scraping the text content of reviews from the webpage.



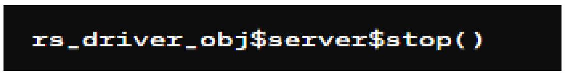
```
all_reviews = remDr$findElement(using = "xpath", "//*[@id='yDmH0d']/div[4]/div[2]/div
reviews = all_reviews$findElement(using = "xpath","//div[@class='h3YV2d']")$getElemen
```

6. Scraping the text content of reviews from the webpage:



```
all_reviews = remDr$findElement(using = "xpath", "//*[@id='yDmH0d']/div[4]/div[2]/div
reviews = all_reviews$findElement(using = "xpath","//div[@class='h3YV2d']")$getElemen
```

7. Closing Web Browser: After data collection, we will close the web browser instance.

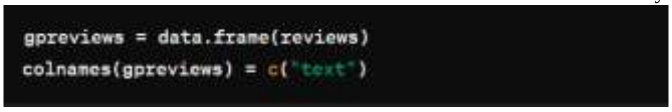


```
rs_driver_obj$server$stop()
```

B. Data Preprocessing and Sentiment Analysis The collected reviews underwent preprocessing and sentiment analysis using the tidytext package. The following steps were involved:

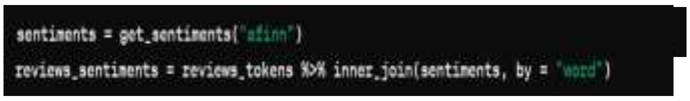
1. Data Preparation: Convert the scraped content into a data frame for further processing.

2. Tokenization: Each review text was tokenized into individual words for analysis.



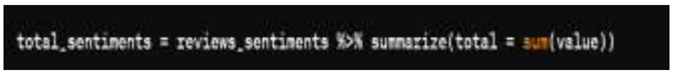
```
gpreviews = data.frame(reviews)
colnames(gpreviews) = c("text")
```

3. Sentiment Assigning: Sentiment scores were assigned to each word using the AFINN lexicon.



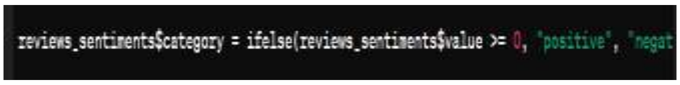
```
sentiments = get_sentiments("afinn")
reviews_sentiments = reviews_tokens %>% inner_join(sentiments, by = "word")
```

4. Total Sentiment Calculation: The overall sentiment of the reviews was computed by summing up the sentiment scores.



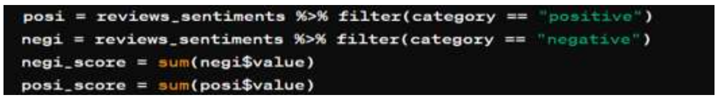
```
total_sentiments = reviews_sentiments %>% summarize(total = sum(value))
```

5. Categorization: Words were categorized into positive and negative sentiments based on their scores.



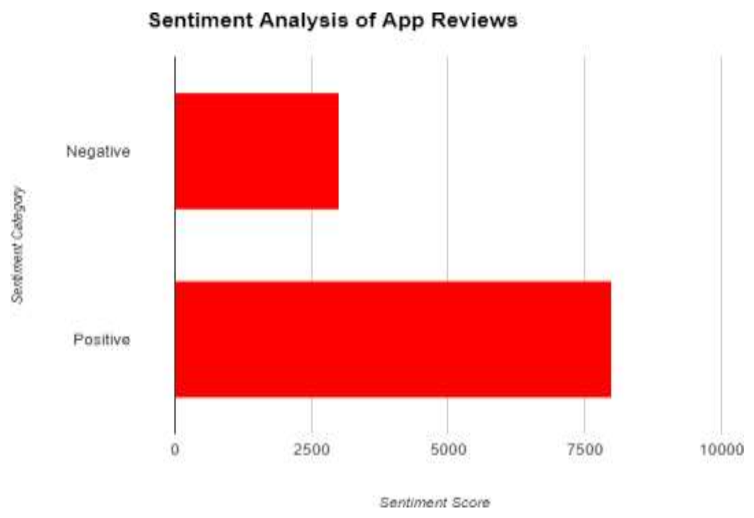
```
reviews_sentiments$category = ifelse(reviews_sentiments$value >= 0, "positive", "negat
```

6. Sentiment Score Aggregation: Calculating the total positive and negative sentiment scores.



```
posi = reviews_sentiments %>% filter(category == "positive")
negi = reviews_sentiments %>% filter(category == "negative")
negi_score = sum(negi$value)
posi_score = sum(posi$value)
```

C. Visualization To visually represent the sentiment analysis results, we employed a bar plot. The positive and negative sentiment scores were depicted as follows:



IV. CONCLUSION

In conclusion, our analysis of user reviews for the applications from the Google Play Store has provided valuable insights into user sentiment and feedback. Despite a mixture of positive and negative sentiments expressed by users. Moving forward, developers can leverage these insights to enhance the overall user experience, foster community engagement, and drive continued app growth and user satisfaction. While acknowledging the limitations inherent in our study, such as reliance on a single data source and potential biases in user-generated content, our findings offer actionable recommendations for developers to refine and evolve the application in line with user expectations and preferences.

REFERENCES

[1] Bernardo, F., & Santos, M. (2020). Sentiment Analysis in Google Play Reviews: A Survey. In 2020 International Conference on Electronics, Communications and Computers (CONIELECOMP) (pp. 1-6). IEEE.

[2] Liu, B. (2012). Sentiment analysis and opinion mining. Synthesis Lectures on Human Language Technologies, 5(1), 1-167.

[3] Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. Foundations and Trends® in Information Retrieval, 2(1-2), 1-135.

[4] Ravi, K., & Ravi, V. (2015). A survey on opinion mining and sentiment analysis: tasks, approaches and applications. Knowledge-Based Systems, 89, 14-46.

[5] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 conference on empirical methods in natural language processing (pp. 1631-1642).

[6] Garg, N., Schiebinger, L., Jurafsky, D., & Zou, J. (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. Proceedings of the National Academy of Sciences, 115(16), E3635- E3644.

[7] Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., ... & Yutani, H. (2019). Welcome to the tidyverse. Journal of Open Source Software, 4(43), 1686.

[8] Xie, Y. (2020). knitr: A general-purpose package for dynamic report generation in R. R package version 1.29.

[9] Chang, W., Cheng, J., Allaire, J., Xie, Y., & McPherson, J. (2020). shiny: Web Application Framework for R. R package version 1.5.0.