Final Project Report: Review Insights

Developers of mobile applications have to publish their apps on app stores in order for users to quickly and easily find and install their apps. These app stores let users leave reviews, where they rate the application on a spectrum from 1 star to 5 stars, with 5 being considered the best review. These reviews also include a text entry, allowing users to comment on the app or leave a rationale as to why they rated the application the way they did. It is common practice for users to complain about bugs or weird application behavior in the reviews, which means that developers can utilize these to gauge the general consensus on what users think of the app. The data that can be gathered from reviews can be invaluable for a development team, they can get production feedback from real users on real bugs, as well as potential quality-of-life improvements which can be made into user stories for future sprints.

The problem is that apps, especially popular ones, can get several hundred to thousands of reviews daily. Simply reading these reviews every day, as they came in, would be a herculean task, which would require countless man-hours of effort. This renders the task of manually reading reviews impractical. Review Insights is a tool developed in order to alleviate the task of sifting through thousands upon thousands of reviews, to hopefully highlight the most useful entries, which developers can then analyze to gather invaluable data. The tool automatically singles out the most-positive and most-negative reviews from the batch of 1000 newest reviews for an application of the user's choosing.

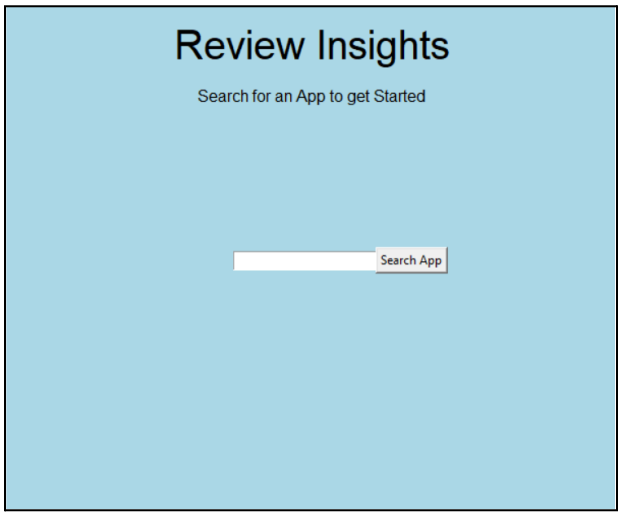

Figure 1. Review Insight's Home Screen

Review Insights, which was developed using Python, utilizes the Tkinter library to build a graphical user interface for the usage of the application. The first screen the user encounters when opening the app is the home screen, seen in Figure 1, which contains the name of the application as well as a textbox entry and a prompt to enter the name of an application. After being supplied with a valid application name, and after the user presses the search button, Review Insights utilizes the Google-Play-Scraper project, maintained by PlanB, to scrape the top 5 results that match the query string from the Google Play Store. The query string equals the app name provided in the homescreen's text-box entry.

The next screen the user is presented with, called the results screen, shows these top 5 results, shown one after another, accompanied with the picture of the application on the Google Play Store, and a button labeled 'select'. This screen exists simply to let the user select the correct application to analyze, in the case that there exists more than one application with a similar name. In the backend, each of the results shown exist within the code as objects from a class called AppResults. The AppResults class's purpose is twofold: firstly, the objects of this class create, within the results screen, their desired 'result rectangle', which contains within it

the application's name, the image, and the button to select the app for analysis. The second purpose of the class is to, once an application is selected, pass the correct app-id to get_reviews() function, and then once the web scraper gathers the correct reviews, to call the summarize() and analyze() functions and display the correct information on the next screen, which should be the App Analysis screen. The AppResults class is also in charge of the functions for the buttons which toggle between seeing reviews and summaries for good results, bad results, and interesting results (what constitutes an interesting result will be explained later).

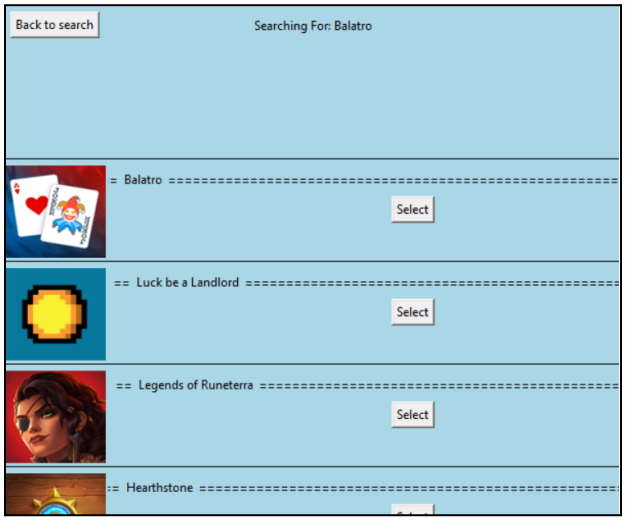

Figure 2. Review Insight's Results Screen

The get_reviews() function works similarly to the function used to get the top 5 results for a given query string, which is to say, using the Google-Play-Scraper. The analyze() function works by utilizing the Natural Language Toolkit (NLTK)'s SentimentIntensityAnalyzer. This tool takes a snippet of code and analyzes it to produce three individual scores: a positivity score, a negativity score, and a neutrality score. All of the reviews for the application are run through the analyzer. Then, the function called good_bad_interst_split() takes the 5 reviews with the highest positive score and the 5 reviews with the highest negative score, and places these into 2 respective pandas dataframes. For the interesting reviews, the code first creates two new dataframes, one with all 5-star reviews, sorted by highest *negative* score and one with all 1-star reviews, sorted by highest *positive* score. The two dataframes are merged and the first 5 results are shown to the user. This process is done so that the user can see results where the sentiment analysis did not match the review's score. There is usually a reason why there is a mismatch between the given star-score and what the sentiment analyzer predicted, so the rationale behind presenting these reviews as 'interesting' is to give the user reviews which were particularly hard for the analyzer to pin down. In my limited testing, these 'interesting' results do indeed seem to highlight, with somewhat regular accuracy, reviews which could be helpful to the application's developers. The summarize() function works by concatenating the top 15 reviews (only the top 5 reviews are shown to the user), and running these reviews through Facebook's BART summarizer model through a HuggingFace pipeline. The resulting summarized text is shown to the user above the reviews.

To highlight the results from this project, and see how Review Insights can be used to quickly gather valuable insights from the reviews of an application, the following example is presented. Balatro is a poker rouge-like video game released in 2024, which was developed by a single person for PCs, and which was very recently ported to mobile, meaning the game was re-factored to be able to work on mobile-phone architecture, and subsequently released on the Google Play Store. Since this game was very recently released, and it was not developed by a large team, it seems like the perfect candidate for testing Review Insights, to see if this project can reveal bugs quickly. When the user searches for Balatro, they will be met with the screen in

Figure 2. If the user then selects the first result to analyze, the results should be similar to the ones seen in Figure 3.

By default, when a user first sees the App Analysis screen, the positive results will be displayed, and the user should see the summary at the top and the highest ranked reviews by positivity score. While these results can be interesting, as they can shine a light on what an app is doing correctly, for the most part, the reviews of interest will be located in the other two tabs: 'Negative Results' and 'Interesting Results'.

Navigating to the negative results, the first three reviews already show the benefit that Review Insights can bring to developers. The first result asks for an option to confirm discards (a mechanic within the game), this review could easily turn into a future user story for a development sprint. The other two reviews both point to different bugs that the development team might or might not be aware of. The review labeled '2)' points to a bug in the game sync/game-save code, while the third review points to a battery drain problem on certain devices. Both of these comments could be potential bugs which could be investigated by the team and could be used to improve the application moving forward.

Moving to the 'Interesting Results' tab also proves fruitful for this app. The longest review on this tab shows that a specific user was dealing with a bug where beating the 'purple stake' (a level of the game) did not unlock the subsequent stake/level. Once again, Review Insights was able to quickly uncover a real bug that a user was experiencing, providing the development team with an actionable item that they can then use to make their application better.

Overall, Review Insights can be used to quickly gather useful information on any application on the Google Play Store, helping development teams find and fix bugs.

Figure 3. Review Insight's App Analysis Screen with Balatro

Figure 4. Balatro's Negative Reviews Analysis

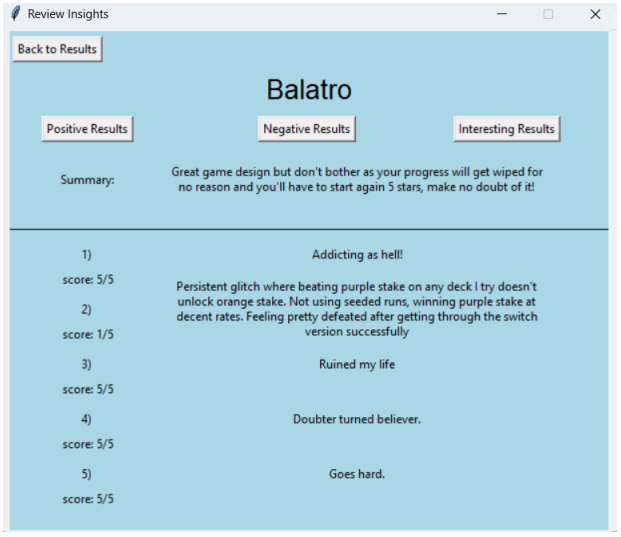

Figure 5. Balatro's Interesting Reviews Analysis

Team Information:
I completed this project on my own.

GitHub Link:
https://github.com/alvaraji/ReviewInsights

Bibliography:

1. Hasija , S. (2021, July 16). *Play store app reviews scrapper (daily update)*. Kaggle. https://www.kaggle.com/code/odins0n/play-store-app-reviews-scrapper-daily-update
2. Mulla, R. (2022, May 5). *Sentiment Analysis Python 🤗 [youtube tutorial]*. Kaggle. https://www.kaggle.com/code/robikscube/sentiment-analysis-python-youtube-tutorial
3. Chaumond, J, et. al. (2024, February 13). *Facebook/Bart-Large-CNN · hugging face.* AI at Meta · Hugging Face. https://huggingface.co/facebook/bart-large-cnn
4. Mingyu, J. (2024, June 6). *Google-play-scraper.* PyPI. https://pypi.org/project/google-play-scraper/
5. NLTK. (2023, January 2). *nltk.sentiment.SentimentIntensityAnalyzer.* https://www.nltk.org/api/nltk.sentiment.SentimentIntensityAnalyzer.html?highlight=sentimentintensity