Android apps and user feedback: a dataset for software evolution and quality improvement

This paper by Giovanni Grano and colleagues (2017) introduces a large, publicly available dataset designed to help researchers understand how Android applications evolve and how user feedback relates to software quality. The dataset combines user reviews, version history, and code quality metrics for 395 open-source Android apps across 23 categories from the F-Droid repository, paired with reviews extracted from Google Play. Altogether, it includes about 288,000 reviews and 629 app versions, along with more than 450,000 labeled user feedback sentences.

The motivation behind the project comes from the observation that mobile app stores like Google Play contain massive amounts of user data that can guide software maintenance and improvement. Developers frequently rely on user reviews to identify bugs, request new features, or understand user satisfaction levels. However, app stores don't organize or filter this information effectively, making it difficult to connect user feedback to specific code changes or versions. The authors address this gap by linking reviews directly to app versions and by computing code quality metrics to show how software evolves alongside user opinions.

To build the dataset, the authors used web crawlers and automated scripts to collect apps and reviews. Apps that hadn't been updated since 2014 or weren't found on Google Play were excluded. The team then linked reviews to specific app versions based on the release and posting dates. Each review was broken into individual sentences and labeled using a two-dimensional classification model called the URM taxonomy, which categorizes feedback by (1) intention: such as feature requests, bug reports, or information sharing, and (2) topic: like GUI, functionality, performance, or security.

For the technical side, the researchers decompiled each app's APK using apktool and analyzed the code to compute 22 quality metrics and detect 8 types of code smells, such as "Blob Class" or "Long Method." They also used Paprika, a static analysis tool, to automatically identify object-oriented and Android-specific anti-patterns. This detailed static analysis gives researchers a consistent way to compare code quality across app versions.

The paper also provides statistical breakdowns of the dataset. On average, each app version had around 458 reviews, and each review consisted of about 1.5 sentences. Common review topics included general app feedback, feature functionality, and GUI issues. The "Tools" category alone accounted for over half the total reviews.

Finally, the authors released the dataset in both relational database and CSV formats through GitHub, including entity-relationship diagrams and detailed documentation. They encourage others to use it for studies that explore relationships between user feedback, software quality, and market success.

The dataset stands out for bringing together three kinds of data, user reviews, app versioning, and code quality metrics, in a single, well-structured collection. Earlier datasets usually focused on just one of these aspects: some gathered app store reviews, while others concentrated on code analysis. By merging both sides, this work enables new types of research questions, such as how certain code smells correlate with negative user feedback, or how user suggestions influence future releases.

In essence, the paper doesn't present new algorithms or experiments but instead contributes a valuable research infrastructure; a foundation that other researchers can use for empirical studies on mobile software evolution.

This paper succeeds in delivering a well-organized, practical resource for both researchers and developers. It demonstrates strong technical depth in how the authors

combined data collection, natural language processing, and static code analysis. The decision to use existing, proven classifiers like ARDOC and SURF was smart as it ensures that the dataset's labeling is reliable without reinventing methods. Also, making everything publicly available through GitHub supports transparency and reproducibility, which is crucial in modern software engineering.

That said, the paper's focus is almost entirely on data construction rather than applying the dataset to real analysis. While it hints at many future research directions like studying how code quality affects user ratings, it doesn't demonstrate any actual correlations or predictive insights. As readers, we would have appreciated a short case study showing how the dataset can reveal actionable findings, even if preliminary.

Another limitation is that the dataset is restricted to open-source apps from F-Droid, which represent only a small and possibly higher-quality subset of the Android ecosystem. Many commercial or mainstream apps behave differently in terms of user engagement and update frequency. This might limit how broadly the findings generalize.

On the other hand, this limitation also makes the dataset manageable and ethically clean since open-source apps allow code inspection. The choice ensures transparency and reproducibility; two strong merits that commercial app data can't easily offer.

We also liked that the authors discuss review intentions and topics instead of treating all feedback equally. The URM taxonomy makes the dataset much richer, allowing researchers to separate, for example, feature requests from complaints. That's essential if we ever want automated systems to prioritize developer actions based on user sentiment. (we know this for a fact given the nature of our senior project)

One more strength is the linking of user reviews to specific app versions, a detail often ignored in previous datasets. This connection opens the door to interdisciplinary studies,

letting researchers see whether user satisfaction improves after certain code changes or refactorings.

From a personal standpoint, we find the dataset concept inspiring because it merges software engineering and human feedback analysis, two areas that usually stay separate. It reminds us that improving software isn't just about writing cleaner code; it's also about listening to users and quantifying how development decisions affect real experiences. We agree with the authors' underlying philosophy: user reviews are not noise, they are valuable signals for guiding software evolution.

If we had to suggest one improvement, it would be integrating sentiment analysis trends or time-based feedback changesdirectly into the dataset. Seeing how user sentiment evolves over releases could make the data even more useful for predicting app success or developer responsiveness.

Overall, this paper contributes a well-designed dataset that bridges the gap between software quality metrics and real-world user feedback. It's less about presenting new theories and more about enabling future discoveries. The authors show that combining static code analysis with user review mining can create a deeper, more dynamic understanding of software evolution.

Even though it's primarily a data paper, the work stands as a strong foundation for future research in mobile app analytics, software maintenance, and developer decision-making. We found it both informative and forward-looking; an excellent example of how empirical data can connect developers and users in meaningful ways.