QuickReview: A Novel Data-Driven Mobile User Interface for Reporting Problematic App Features

Tavita Su'a, Sherlock A. Licorish, Bastin Tony Roy Savarimuthu, Tobias Langlotz
Department of Information Science, University of Otago
Dunedin, New Zealand
tavitasiona.sua@gmail.com, {sherlock.licorish, tony.savarimuthu, tobias.langlotz}@otago.ac.nz

ABSTRACT
User-reviews of mobile applications provide information that benefits other users and developers. Even though reviews contain feedback about an app's performance and problematic features, users and app developers need to spend considerable effort reading and analyzing the feedback provided. In this work, we introduce and evaluate QuickReview, an intelligent user interface for reporting problematic app features. Preliminary user evaluations show that QuickReview facilitates users to add reviews swiftly with ease, and also helps developers with quick interpretation of submitted reviews by presenting a ranked list of commonly reported features.

Author Keywords
User Interface; App Reviews; Android; Mobile Devices, Data Driven; Intelligent User Interfaces

ACM Classification Keywords
H.5.2. User Interfaces; H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous;

INTRODUCTION
Contemporary app marketplaces provide users and app developers a feedback channel, in the form of app reviews and star ratings [2]. To this end, users' reviews are essential to the lifecycle of an app in fulfilling two distinct purposes. App reviews: (1) allow *developers* to identify bugs and problematic features and, (2) enlighten new *users* about an app's weaknesses, and facilitate users' purchasing and downloading decisions [1]. However, for both these parties, the review process poses many challenges [3]. App developers of popular apps need to read hundreds of app reviews to identify common problems. In addition, users are not able to endorse or confirm previous reviews, but instead must write new reviews if the need arises [9]. This increases the number of reviews that need manual text-interpretation, even though only around 35% of app reviews have been shown to offer information that can directly help developers to improve their apps [2].

To bridge this gap, this work explores both **the feasibility and usefulness of developing an intelligent and adaptive user interface, QuickReview**. QuickReview is motivated by Von Reischach et al. who investigated the differences in reviews and rating on mobile devices when compared to web-based systems, confirming that mobile users prefer *less but aggregated* product information [4]. This holds true for both entering reviews and browsing reviews. QuickReview provides a data-driven interface that adapts based on existing reviews and allows users to easily provide feedback on problematic features, while also aggregating the most reported problematic features for app developers and perspective new users.

We provide a review of related work in the following section, before presenting our development of QuickReview. Thereafter, we provide our evaluation setup and the outcomes. We then briefly discuss our findings and provide concluding remarks.

BACKGROUND
Several research groups have looked into related problems of optimizing the presentation of product reviews. Jin et al. for example, proposed a solution that focuses on summarizing reviews for better comparison of two product candidates by extracting attribute-value pairs from longer reviews and presenting them in a one-page summary view [5]. However, their work focuses on comparing two different ways of presenting information, and not on an interface for providing and aggregating reviews, which forms the focus of our work. Dong et al., focused on developing an intelligent review assistant that recommends topics for writing better reviews on the web that contains more relevant facts [6, 7]. There are also works that do not focus on writing better reviews, but aim to aggregate and analyse reviews to extract meaningful information. For example, Liu et al. [8] used text mining and sentiment analysis tools to extract and identify user sentiments of specific app features. Their approach to analysing app reviews is similar to that conducted by Patel et al. [9], whose app analytics were adapted in extracting the problematic features for the app(s) being reviewed in the current study. A recent study by Chen et al. [10] proposed AR-Miner for mining app reviews to extract the most informative reviews for app developers. This study

IUI 2017, March 13 - 16, 2017, Limassol, Cyprus
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-4348-0/17/03…$15.00
DOI: http://dx.doi.org/10.1145/3025171.3025178

showcases the advancement in app review mining and allows features to be identified for further consideration, in supporting app improvements. The work in [24] provides auto-completion suggestions for a new reviewer based on reviews submitted by other users. However, this involves all users to manually enter their reviews, which this work aims to minimize, amidst other goals.

While works reported in [9] and [10] have largely supported data extraction, our work goes one step further by presenting summarised outcomes within a mobile interface. This summarised data is subsequently used for generating new reviews (reports about problematic app features). To the best of our knowledge, this work is the first to present **a data-driven mobile interface to improve the quality and structure of app reviews**. In addition, preliminary evaluations suggest that our intelligent app, QuickReview, could be of utility to users and developers.

QUICKREVIEW DEVELOPMENT

We anticipated that an intuitive, data-driven interface needs to meet three aspects of mobile user interfaces. First, it should be able to eliminate the need for extensive human analysis that is currently required to process voluminous reviews, while at the same time supporting the users of an app to generate a new review with ease (i.e., *enable better usability when compared to a traditional system*) [21]. Second, beyond aiding users' in generating reviews, an intuitive review interface should demand *minimal cognitive load*, in taking account of varying mobile contexts often requiring users' attention [22]. Third, the *performance* of the data-driven interface should be superior to that of traditional systems used for capturing users' feedback [23]. We considered these criteria to guide the development of our intelligent and adaptive user interface.

Review Extraction and Processing

We first employed natural language processing (NLP) techniques to the reviews extracted from Google Play. Our goal was to process reviews to identify the problematic features reported in the reviews and the nature of issues (e.g., "GPS feature being slow", where *GPS* is the feature and *slowness* is the issue). In order to focus on features that are problematic, we extracted all reviews with negative emotions such as anger, sadness and fear, which signal discontent of users [11]. We used the LIWC tool dictionary to inform our negative words cohort, which contains 431 negative words [12]. Noun terms in unstructured text reflect the main concepts in the subject of a clause. From a part-of-speech (POS) perspective, nouns are indeed reflective of specific objects or things. From a linguistic perspective, nouns often form the subjects and objects of clauses or verb phrases. Hence, nouns are the features that are deemed problematic and the verbs are the issues. These and other understandings have been embedded as rules in natural language processing (NLP) tools, such as the Stanford parser which performs POS tagging [13]. We incorporated the Stanford API in our toolset to enable us to extract noun phrases (features) from reviews, before counting the frequency of each noun as a unigram (e.g., if "SMS" appeared at least once in each of 20 reviews, our app would then output SMS = 20). The ranking of words in this manner draws from computational linguistics, and is referred to as n-gram analysis. The n-gram is defined as a continuous sequence of n words in length that is extracted from a larger stream of elements [14]. We extracted the syntactic relations between pairs of features (nouns) and issues (verbs) in each request by providing counts of these noun-verb pairs in the reviews. For example, if one review reads "the *Search* feature *freezes* every five minutes", and another "the *Search* feature always *freezes*", our output would be Search-freezes = 2. We aggregated these feature-issue occurrences as input for the QuickReview interface, where the counts was used for ranking.

A Data-Driven User Interface (QuickReview)

Outputs extracted from reviews as described in the previous section were used to populate QuickReview's user interface as an actionable graphical element (with buttons and check boxes), which allows users to select the features and issues they wish to include in their report (see Figures 1b and 1c). We anticipate that this would reduce the need for entering detailed comments when compared to traditional review interfaces (see Figure 1a). Given that some apps may have little reviews, and thus, the text mining process may not be relevant in this context, QuickReview also allows users to optionally enter descriptive reviews expressing their personal opinion if needed (see Figure 1d). Thus, a review in QuickReview will contain a set of problematic features and issues selected by the user (e.g., *Battery* and *drain*) and/or an optional descriptive review. We were careful to design QuickReview interface with consideration for the limited screen size and input methods associated with mobile devices, also conforming to the Android design guidelines. An iterative approach to user interface development was adopted, with the final interface using a minimalistic design (exposing information only essential for the current user task).

We first developed a replica of the Google Play review interface as a basis for comparison with QuickReview (see Figure 1a). Using the Google Play review interface, users are able to add a textual review and provide a star rating. Thereafter, we designed and developed QuickReview (see Figures 1b – 1d). In what follows, we discuss QuickReview design choices and implications for the overview screen (Figure 1a) before the comparative presentation of the data-driven interface with the Google Play interface.

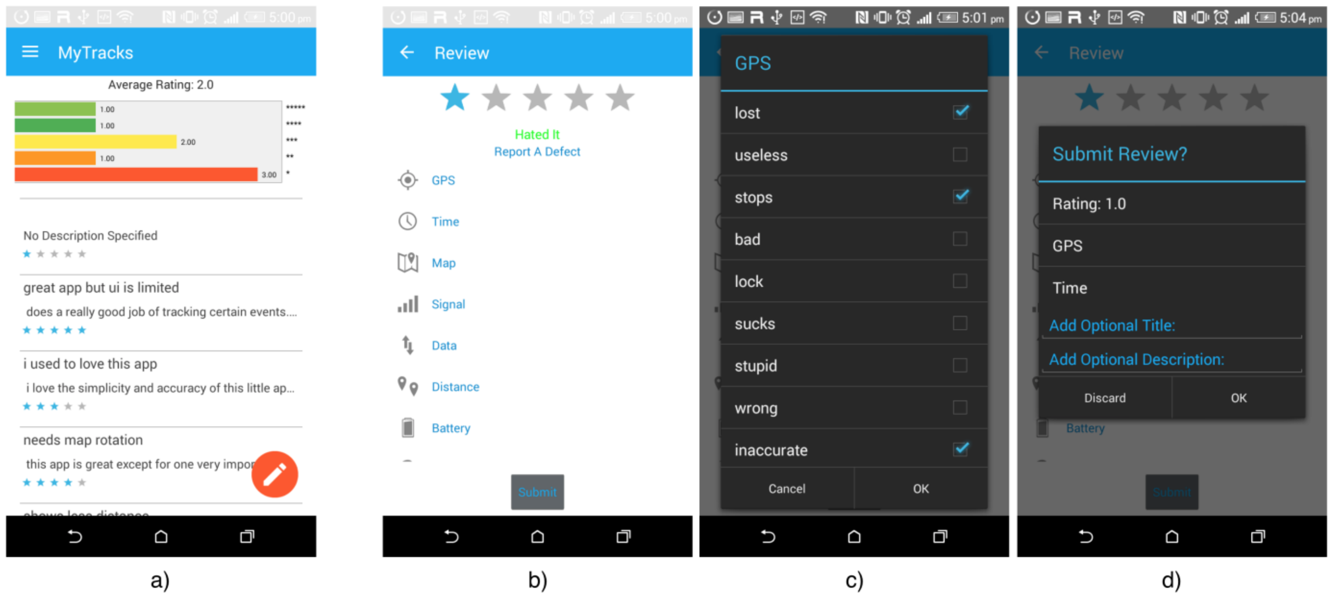

Figure 1. Traditional app review system and QuickReview. a) App review interface as used in the Google Play Store. b) Proposed QuickReview interface extending traditional interface by presenting problematic features (e.g. GPS and Time) extracted by mining the existing text reviews and displaying them. c) Selecting a feature (here GPS) displays corresponding issues, also extracted using data mining, for this specific feature such as "lost" and "stops". Users can confirm one or multiple issues for the selected feature without typing lengthy reviews. d) Users can still provide additional information via text comments.

The overview screen shown on the left of Figure 1(Figure 1a) uses a Master/Detail structure where app reviews are shown without cluttering the main screen based on design principles reported in other works [15, 16]. Also, the overview screen presents a single clickable button (indicated by the Orange oval with an edit-icon) that presents a clear and consistent action choice that minimizes the cognitive load on users [17-19].

When the button is selected, the QuickReview interface showcases a vertical list of selectable elements, ranked based on the most commonly reported features about the specific app being reviewed, extracted during the automatic review analysis (see Figure 1b). The use of specific icons for the features provides easy recognition of the features, a shared principle found in many studies (e.g. [19]). Touching a certain feature (e.g., GPS) shows a different vertical list highlighting the top ten automatically extracted issues (i.e. issues co-occurring with the currently selected feature (see Figure 1c)). Again, a Master/Detail approach was applied to this screen to separate the detailed issues that users have associated with each identified feature in the list. These issues are also arranged in descending order based on occurrence in the reviews. The user can select the issues they would like to report.

On clicking the OK button on Figure 1c users are taken to the submit screen (see Figure 1d) which presents candidate features and issues selected by the user (also confirming previous users' complaints). The reason for presenting a list of previous users' problematic features and the corresponding issues is to reduce the choices made available upfront to the user, thus reducing their cognitive load (i.e., a user's memory about a feature can be kindled through a list of previous problems reported in reviews; developers are also able to browse options in Figures 1b and 1c to explore users' feedback about app features). Having provided that, the user is also able to provide a textual response via the app review summary screen (see Figure 1d). This screen allows users to inspect their app review before submission, and enables them to easily change any aspect of their review (rating, identified features/issues, title or description).

It should be noted that the QuickReview app was designed for a top-down screen structure to avoid potential information overload by arranging the functionalities using a multilevel hierarchy [20], thus exposing users only to information that is necessary for each intermediate step. The review interface on Google Play Store was developed (which has the same overview screen of Figure 1a) in order to facilitate comparisons with QuickReview. Upon selecting the review button in the overview screen, textual comments and the star rating can be provided by the user (not shown here due to space limitation).

QUICKREVIEW EVALUATION SETUP

A user study was conducted to evaluate QuickReview in comparison to the existing interface on Google Play Store. The study used a randomized trial as part of a within-subject design comparing the Google Play Store review interface (now referred to as *GP*) against our data-driven interface, QuickReview (now referred to as *QR*). For both cases we populated the interfaces with data from existing app reviews (4500 reviews) for the app MyTracks[1] as the dataset was made available to us and the described app showed enough weaknesses that were worth analysing. After a short introduction to the study and background questionnaires, the participants were presented with two scenarios (one simple and one complex scenario) based on actual reported problems with the MyTracks app. For each scenario the participants had to report a problem using each of the review interfaces (GP and QR). Note GP requires the provision of a text-summary of the problem while QR provides an intelligent and adaptive interface.

Upon studying both scenarios and completing their tasks, participants were then asked to answer two questionnaires on usability and cognitive load. Usability was measured using a modified System Usability Scale (SUS) [21]. Four questionnaire items (questions 2, 5, 6 and 7) were omitted as they were not applicable to app evaluations resulting in 6 questions. The resulting usability scores from the SUS were calculated from the questionnaire items using the weighted calculation, ranging from 0 – 60 instead of 0 – 100 as described by Brooke [21]. Cognitive load was measured using the standard NASA-Task Load Index (TLX) [22] questionnaire. Finally, performance of the two apps was measured by recording the time taken by the users to complete app reviews using both interfaces, and we also recorded task completion rate and error rate. Twenty (20) participants (age 18 to 24 years, 13 females and 7 males) evaluated QuickReview, with all participants using a Samsung Galaxy S3 Android smartphone.

QUICKREVIEW EVALUATION OUTCOMES

An alpha level of 0.05 was applied for all statistical tests, and our outcomes are presented below.

Usability: Our outcomes show that QR had a higher mean (M) usability score than the GP interface (49 versus 44; with standard deviation (SD) for QR=12.5 and GP=13.9). However, an independent samples *t*-test shows that these results were not significantly different ($p>0.05$).

Cognitive Load: Six workload measures were used to examine cognitive load: mental demand, physical demand, time pressure, effort expended, performance and frustration. An overall cognitive load score was calculated, combining these scores into a single mean score for each participant (M: QR=23.5, GP=35.3; SD: QR=18.9, GP=26.2). These measures indicate that QR required less cognitive load when adding reviews than GP (mean difference of 11.8). An independent samples *t*-test conducted showed no significant difference ($p>0.05$). Follow up tests for each of the cognitive dimensions were also not significant.

Performance: As noted above, we initially considered measures for time taken for task completion, task completion rate (whether a task was successfully completed), and error rate (whether a participant misused or misunderstood features of the app) when testing the interfaces of GP and QR. However, task completion rate and error rate did not produce any data points, as all evaluations were completed successfully without errors. Therefore, only time for task completion was considered when comparing performance for QR and GP. These distributions violated normality, and thus, a Mann-Whitney U non-parametric test was conducted which confirmed statistically significant differences in performance when conducting reviews using GP ($M = 67.7$, $SD = 19.6$) and QR ($M = 50.3$, $SD = 20.10$), with the process being much faster on QR, $z = -2.37$, $p = 0.01$. The effect size associated with this finding, $d = 0.88$, was found to exceed Cohen's size convention for a large effect size ($d = 0.8$).

DISCUSSION AND CONCLUSION

We examined the **feasibility and usefulness of developing an intelligent and adaptive user interface, QuickReview**. While two aspects our results were not significant, QuickReview recorded *higher usability score* than the current Google Play interface for adding reviews. In addition, evaluation outcomes show that QuickReview demanded *less cognitive workload*. The utility of QuickReview in these aspects is exemplified through participants' written feedback. One noted that the new system was "easy to use and understand" and the other noted it "didn't take too much physical and mental effort". These findings are particularly satisfying given that the lack of familiarity may have moderated our results somewhat, as all of the users were familiar with the old review system on Google Play, while only seeing and using QuickReview for the first time when introduced in the evaluation. Psychologists have established that cued recall (i.e., prompting based recall of QuickReview) is faster than free recall (i.e., recollecting and then writing reviews as done via Google Play). We believe that this may have been reflected in the decrease in cognitive load for all the six cognitive load factors for QuickReview [19]. This assessment may be particularly valid given the results for performance outcomes, which also show that *users were able to log reviews much faster* when using QuickReview than the Google Play interface. This confirms the relevance of *limiting time overhead* in mobile interface design [23].

QuickReview could also be useful for app developers. The populated list of features and corresponding issues are ranked and presented such that improvement opportunities could be quickly identified by the developers. That said, while these initial outcomes are encouraging, a power analysis using the GPower indicated that 336 evaluators were needed to detect large effects ($d = 0.8$) with 95% power using an independent samples *t*-test with an alpha value at 0.05. We thus plan to extend our evaluations, both in terms of respondents and including reviews for a wider range of apps. We also plan to extend QuickReview to include positive aspects of apps, showing how key features satisfied users.

[1]https://play.google.com/store/apps/details?id=com.google.android.maps.mytracks&hl=en

REFERENCES

1. Judith A Chevalier and Dina Mayzlin. 2006. The effect of word of mouth on sales: Online book reviews. *Journal of marketing research*, 43 (3). 345-354.

2. Chen, Jialiu Lin, Steven CH Hoi, Xiaokui Xiao and Boshen Zhang. 2014. AR-Miner: mining informative reviews for developers from mobile app marketplace. In *Proc. of the ICSE conference,* ACM, 767-778.

3. Bin Fu, Jialiu Lin, Lei Li, Christos Faloutsos, Jason Hong and Norman Sadeh. 2013. Why people hate your app: Making sense of user feedback in a mobile app store. In *Proc. of SIGKDD conference*, 1276-1284.

4. Felix Von Reischach, Erica Dubach, Florian Michahelles and Albrecht Schmidt. 2010. An evaluation of product review modalities for mobile phones. In *Proc. of the 12th international conference on Human computer interaction with mobile devices and services*, ACM, 199-208.

5. Haojian Jin, Tetsuya Sakai and Koji Yatani. 2014. ReviewCollage: a mobile interface for direct comparison using online reviews. In *Proc. of the 16th international conference on Human-computer interaction with mobile devices & services*, ACM, 349-358.

6. Ruihai Dong, Kevin McCarthy, Michael O'Mahony, Markus Schaal and Barry Smyth. 2012. First demonstration of the intelligent reviewer's assistant. In *Proc. of the 2012 ACM international conference on IUI*, ACM, 337-338.

7. Ruihai Dong, Kevin McCarthy, Michael O'Mahony, Markus Schaal and Barry Smyth. 2012. Towards an intelligent reviewer's assistant: recommending topics to help users to write better product reviews. In *Proc. of the 2012 ACM international conference on Intelligent User Interfaces*, ACM, 159-168.

8. Jiawen Liu, Mantosh Kumar Sarkar and Goutam Chakraborty. 2013. Feature-based Sentiment Analysis on Android App Reviews Using SAS® Text Miner and SAS® Sentiment Analysis Studio. In *Proc. of the SAS Global Forum 2013 Conference*.

9. Patel, P., Licorish, S., Savarimuthu, B. T. R. and MacDonell, S. 2016. Studying expectation violations in socio-technical systems - A case study of the mobile app community In *Proc. of the European Conference on Information Systems* (ECIS) (Istanbul, Turkey).

10. Chen, Jialiu Lin, Steven CH Hoi, Xiaokui Xiao and Boshen Zhang. 2014. AR-Miner: mining informative reviews for developers from mobile app marketplace. In *Proc. of the ICSE conference*, ACM, 767-778.

11. Guzman, Emitza, and Walid Maalej. 2014. How do users like this feature? a fine grained sentiment analysis of app reviews. In *Proc. of the RE conference*, 153-162.

12. Pennebaker, J. W., Francis, M. E. and Booth, R. J. Linguistic Inquiry and Word Count. *Mahway: Lawrence Erlbaum Associates*, 71 (2001).

13. Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network, In *Proc. of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology* - Volume 1. Edmonton, Canada: Association for Computational Linguistics, pp. 173-180.

14. Manning, C. D., and Schtze, H. 1991. *Foundations of Statistical Natural Language Processing*. London: MIT Press.

15. Orkut Buyukkokten, Hector Garcia-Molina and Andreas Paepcke. 2001. Seeing the whole in parts: text summarization for web browsing on handheld devices. In *Proc. of the international conference on WWW*, ACM, 652-662.

16. Thanh-Diane Nguyen and Jean Vanderdonckt. 2012. User interface master detail pattern on Android. In *Proc. of the 4th ACM SIGCHI symposium on Engineering interactive computing systems*, ACM, 299-304.

17. Luca Chittaro and Paolo Dal Cin. 2002. Evaluating interface design choices on WAP phones: Navigation and selection. *Personal and Ubiquitous Computing*, 6 (4). 237-244.

18. Rachel Harrison, Derek Flood and David Duce. 2013. Usability of mobile applications: literature review and rationale for a new usability model. *Journal of Interaction Science*, 1 (1). 1-16.

19. M. Negulescu, J. Ruiz, Y. Li and E. Lank. 2012. Tap, swipe, or move: Attentional demands for distracted smartphone input. In *Proc. of International Working Conference on Advanced Visual Interfaces*, AVI 2012, Capri Island, 173-180. 10.1145/2254556.2254589

20. Stephen Brewster. 2002. Overcoming the lack of screen space on mobile computers. *Personal and Ubiquitous Computing*, 6 (3). 188-205.

21. John Brooke. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry*, 189 (194). 4-7.

22. Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. *Advances in psychology*, 52. 139-183.

23. Ivan Poupyrev, Shigeaki Maruyama and Jun Rekimoto. 2002. Ambient touch: designing tactile interfaces for handheld devices. In *Proc. of the 15th annual ACM symposium on User interface software and technology*, ACM, 51-60.

24. Arnold, Kenneth C., Krzysztof Z. Gajos, and Adam T. Kalai. 2016. On Suggesting Phrases vs. Predicting Words for Mobile Text Composition. In *Proc. of the 29th Annual Symposium on User Interface Software and Technology*, ACM, 603-608.