RE-SWOT: From User Feedback to Requirements via Competitor Analysis

Fabiano Dalpiaz[1][0000−0003−4480−3887] and Micaela Parente[2]

[1] RE-Lab, Dept. of Information and Computing Sciences,
Utrecht University, The Netherlands
f.dalpiaz@uu.nl
[2] Scaura B.V., Amsterdam, The Netherlands
micaelagparente@gmail.com

Abstract. [**Context & Motivation**] App store reviews are a rich source for analysts to elicit requirements from user feedback, for they describe bugs to be fixed, requested features, and possible improvements. Product development teams need new techniques that help them make real-time decisions based on user feedback. [**Question/Problem**] Researchers have proposed natural language processing (NLP) techniques for extracting and organizing requirements-relevant knowledge from the reviews for one specific app. However, no attention has been paid to studying whether and how requirements can be identified from competing products. [**Principal ideas/results**] We propose RE-SWOT, a tool-supported method for eliciting requirements from app store reviews through competitor analysis. RE-SWOT combines NLP algorithms with information visualization techniques. We evaluate the usefulness of RE-SWOT with expert product managers from three mobile app companies. [**Contribution**] Our preliminary results show that competitor analysis is a promising path for research that has direct impact on the requirements engineering practice in modern app development companies.

Keywords: Requirements Engineering · SWOT Analysis · Natural Language Processing · Requirements Analytics · CrowdRE.

1 Introduction

User feedback is a precious resource for requirements elicitation [1, 12, 18]. When effectively managed, user involvement may be beneficial for project success [1]. On the contrary, ill-managed user involvement may be harmful, e.g, if excessive effort is required for processing the collected feedback.

Crowd-based Requirements Engineering (CrowdRE) is a recent trend in Requirements Engineering (RE) that studies semi-automated methods to gather and analyze information from a large number of users, ultimately resulting in validated user requirements [11]. Automation, which is a distinguishing feature of CrowdRE, reduces the effort required to cope with high volumes of feedback.

Natural Language Processing (NLP) techniques have been employed in CrowdRE for summarizing and classifying user input into structured knowledge. Many

of the existing approaches process user-generated reviews posted on app stores: Guzman and Maalej [12] automatically extract app features and the associated sentiment, Di Sorbo *et al.* [7] organize reviews according to their intention, and the AR-Miner tool [4] identifies and summarizes the most informative reviews.

Current CrowdRE approaches that analyze app store reviews focus on a single app. A gap exists in the use of competitor analysis to uncover requirements based on an explicit comparison of one app's reviews with those of competing apps. The only work we could identify that considers competitors (see Sec. 2) extracts and compares pairs of sentences for the same feature from multiple reviews [16].

In this paper, we propose RE-SWOT: a tool-supported method for eliciting requirements from user reviews through competitor analysis. The tool combines NLP automation with information visualization techniques, and belongs to the domain of requirements analytics [5]. Our approach is inspired by classic literature in management, as it adapts the Strength-Weakness-Opportunity-Threat (SWOT) analysis framework [14] to the field of RE. By presenting RE-SWOT, we make three contributions to the literature:

- An algorithm that extracts features from the reviews of a set of competing apps, and then generates a SWOT matrix on the basis of the sentiment that the users have expressed toward the identified features;
- An information visualization technique that plots the results of the algorithm in a chart, and helps analysts visually explore the competing apps with the aim of eliciting new requirements;
- A qualitative evaluation of the practical applicability of our approach. After demonstrating our implemented tool to three product managers of different apps, we collect their opinion through follow-up interviews.

Organization. Sec. 2 discusses background and related literature. Sec. 3 details the algorithm for extracting features and classifying them through the SWOT framework. Sec. 4 illustrates our information visualization tool. Sec. 5 reports on the evaluation, while Sec. 6 discusses our findings and presents future work.

2 Related Work

We present the necessary background for this paper in Sec. 2.1, and discuss related approaches in Sec. 2.2.

2.1 Background

Crowd-based Requirements Engineering. The rise of social media platforms has significantly increased the volume of feedback from software users. As a response, the RE community has initiated a shift toward data-driven and user-centered prioritization, planning, and management of requirements [21].

One of the emerging initiatives is Crowd-based Requirements Engineering (CrowdRE), which is defined [11] as *"an umbrella term for automated or semi-automated approaches to gather and analyze information from a crowd to derive*

validated user requirements". In CrowdRE, the considered feedback comes from users who are not bond with the software company.

App store reviews in requirements elicitation. The reviews that are posted in app stores contain diverse types of feedback. Besides functional issues (e.g., "I used to love it, but I can't watch videos anymore!") and non-functional concerns (e.g., usability or performance), the users also comment on not-yet-implemented aspects by requesting improvements or new features [20].

Analyzing app store reviews can lead to a better understanding of how apps are actually used, fast detection of newly introduced bugs [22], and insights from a more diverse range of users [15]. Unfortunately, several challenges exist:

- *Volume:* to cope with the large quantity of reviews, CrowdRE proposes to use NLP tools [11]. However, requirements elicitation is a "fundamentally human activity" [3] and new tools are called upon to "bringing the human into the loop and promoting thinking about the results" [2].
- *Noise:* Chen *et al.* [4] found that only 35.1% of app reviews contain information that can directly help developers improve their apps. Thus, CrowdRE approaches need to use filtering and aggregation of content [22, 4, 23].
- *One-way communication:* app store reviews lack meta-data about users and app usage; moreover, the user-developer communication is unidirectional. As such, the development team cannot reach back the users and ask for clarifications and context information.
- *Conflicting opinions:* reviews often contain conflicting opinions [11, 28, 15, 21]. Classic negotiations mechanisms are inhibited by the unidirectional communication; as a result, prioritization approaches are necessary to weight issues and wishes according to their prevalence and impact [19].

In practice, app development companies depend on community managers [19] for reading and replying to user reviews, often facilitated by tools that provide average ratings, distribution of stars over time, automatic labeling of reviews, etc. However, such tools do not explicitly support the elicitation of new requirements.

Visual Requirements Analytics. Reddivari *et al.* [26] proposed a framework that characterizes the visual requirements analytics process. They argue that, when proposing a requirements visualization, one has to explicitly define many aspects: the user, the goal, the questions to be answered, how to preprocess data, and the visualization type. Furthermore, they explain how requirements analytics tools need to go beyond the mere visualization and rather focus on the *interaction* between analyst and visualization.

2.2 Related literature: mining requirements from app store reviews

Opinion mining approaches are applicable to app store reviews, but extra challenges exist [10] due to (i) the fine-grained reviews that include comments not only on product features, but also specific parts of the user interface, or particular user-app interactions; and (ii) the short length of app store reviews (71

characters on average [10]). While most papers are still exploratory, some key steps of mining requirements from app store reviews are discussed in the following.

Preprocessing reviews. This activity is necessary to reduce noise in the reviews. Typical techniques include stop word removal, stemming, and lemmatization. Unfortunately, none of them delivers perfect accuracy; for example, while removing common English words tends to improve classification accuracy [20], it can hide user intentions (e.g., "should" for a feature request, "but" for a bug). Similarly, lemmatization and stemming are alternative ways for standardizing words with similar meaning, but no clear winner exists [9, 20]. Other preprocessing techniques include removing short reviews [7], matching synonyms, and filtering words having specific POS tags like nouns, adjectives, and verbs [20].

Classifying review content. The goal is to classify reviews according to a given taxonomy. Yang and Liang [31] distinguish between functional and non-functional requirements by searching for keywords that are typically associated with either category. Most techniques in the literature [7] focus on two aspects: (i) *user intention*: the user's goals when writing the review (e.g., reporting a bug vs. requesting a feature); and (ii) *review topic*: the entire app, its interface, or a specific feature. Maalej and Nabil [20] compare various algorithms for classifying review intention as "Bug report", "Feature request", "Rating", or "User experience". Panichella *et al.* [23] employ a different taxonomy: "Feature request", "Problem discovery", "Information seeking" or "Information giving", which is then used by the SURF tool [7] to classify reviews according to both intention and topics: app, GUI, pricing, security, etc.

Extracting features. NLP techniques can automatically extract the features that a review refers to. Harman *et al.* [13] extract features from publicly available app descriptions from an app store. They rely on the informal patterns that developers use to illustrate the main features of an app, like bullet lists. Based on these patterns, groups of commonly occurring co-located words are employed to represent the feature. Guzman and Maalej [12] use word collocations appearing in at least 3 reviews in their fine-grained analysis. Gao *et al.* [9] identify "phrases" derived from bi-grams to prioritize issues for developers.

The SAFE framework [17] improves over previous methods, and it exhibits a precision of 70%, recall of 56% and F1-score of 62%. They do so by defining a list of POS patterns and sentence patterns that are frequently used to describe features, and then applying cosine similarity algorithms.

Summarizing reviews. Algorithms in this category reduce the feedback volume. AR-Miner [4] summarizes reviews by displaying the ten most important topics (groups of features) found, ranked by an importance score called *GroupScore*. Guzman *et al.* [12] use a two-level summary that shows the frequency and sentiment per topic (groups of features) or per feature.

Competitor analysis. Jin *et al.* [16] identify comparable sentences in different reviews. Through feature extraction, sentiment analysis, similarity functions and clustering, they compare the opinions on a topic by analyzing pairs of extracted sentences. The WisCom system [8] enables summarization at the review, app, and market level. These precursory approaches, however, do not provide a systematic method for eliciting requirements through competitor analysis.

3 The RE-SWOT Method

Our method for eliciting requirements from app store reviews is inspired by SWOT analysis, a prominent framework for strategic planning in organizations that gives an overview of how a product or business is positioned, *vis à vis* its external environment [14]. SWOT analysis identifies four types of factors:

- *Strengths:* internal factors that enhance performance. For example, the high loyalty of an organization's employees.
- *Weaknesses:* internal factors that diminish performance. For instance, reliance on too rigid business processes.
- *Opportunities:* external (i.e., outside the organization's reach) enhancers to performance that could be exploited. For example, economic growth.
- *Threats:* external factors that inhibit performance. For example, a large foreign firm joining the domestic market, which increases competition.

The crux of SWOT analysis is that, in order for an organization to improve its competitiveness, it has not only to maximize the internal strengths and minimize its own weaknesses, but also has to be aware and to react quickly and effectively to the changing context by exploiting opportunities and mitigating threats.

We draw a parallel between SWOT analysis and RE for a software product. Consider an app that is distributed through one or more app stores; the requirements for the next releases are expected to leverage its strengths, mitigate the app's weaknesses, exploit market gaps, and imitate the successful features of the competitors. Table 1 shows the RE-SWOT matrix that classifies a feature into one SWOT category by comparing a reference app with a competing app. The *reference app* belongs to the company that executes the RE-SWOT analysis.

Table 1. The RE-SWOT matrix: SWOT analysis adapted to CrowdRE.

App with the feature / Feature performance	*Reference app*	*Competitor app*
Positive and above market average	**Strength:** keep and/or extend the feature	**Threat:** imitate the competitor's feature to survive
Negative and below market average	**Weakness:** fix bugs or improve on a feature's issues	**Opportunity:** launch new feature to exploit an existing gap

The matrix builds on the notion of *feature performance*, a real number in the $[-1, +1]$ range that represents whether the feature implemented in a given app has a prevalently positive (> 0) or negative (< 0) appreciation in the reviews. The same feature may exhibit different performance when implemented in multiple apps. In RE-SWOT, feature performance is calculated automatically from user reviews, as described in Sec. 3.1.

We illustrate the RE-SWOT matrix with some examples. Consider a feature f_1, which is included in a reference app a_r; if the feature performance of f_1 as implemented in a_r is positive and also above the market average for that feature, then f_1 is a *strength* for app a_r. Consider f_2 instead, which is possessed by a competitor app a_c; if the feature performance of f_2 as implemented by a_c is negative and below market average, then f_2 represents an opportunity for a_r.

3.1 Step-by-step method description

We detail the steps of our method (overview in Fig. 1), thus clarifying how the performance of a feature is calculated and how the RE-SWOT matrix is built.

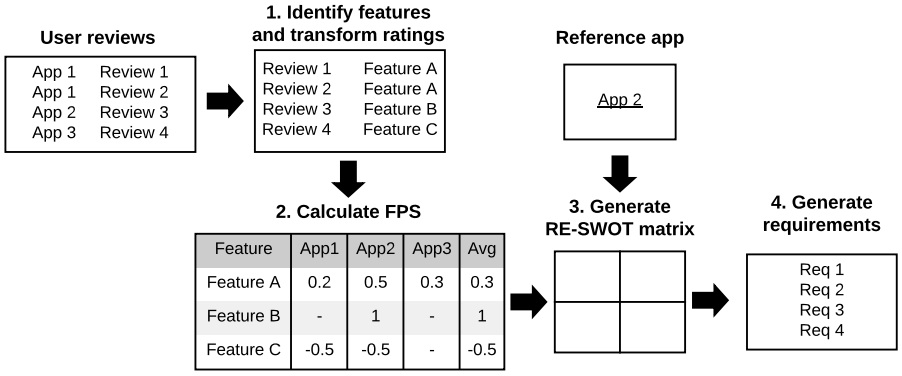

Fig. 1. The RE-SWOT method: an overview.

Step 1: Identify features and transform ratings. App features are identified from the user reviews. To do so, we employ NLP techniques for feature identification, see Sec. 4.1 for the deatils. Furthermore, the original user ratings (a natural number between 1 and 5) are mapped to the $[-2, +2]$ integer scale in which three stars becomes the neutral score (0).

Step 2: Calculate FPS per feature. Given a set of apps $A = \{a_1, a_2, \ldots, a_m\}$ and a set of features $F = \{f_1, f_2, \ldots, f_n\}$, we define the feature performance score (FPS) of app a_i in relation to feature f_j as per Equation 1:

$$FPS_{i,j} = \frac{S_{i,j} \cdot V_{i,j}}{\sum_{i=1}^{m} |S_{i,j} \cdot V_{i,j}|} \tag{1}$$

- $S_{i,j}$ represents the user sentiment for feature f_j from app a_i: the sum of the transformed user ratings given to the reviews mentioning the feature, divided by the maximum possible sum. For instance, if a feature is mentioned in two 5-star reviews and one 2-star review, the feature sentiment score for that feature corresponds to $(2 + 2 - 1)/(2 + 2 + 2) = +0.5$.
- $V_{i,j}$ is the feature volume for feature f_j from app a_i: the number of user reviews from app a_i that mention feature f_j. Take an app with 2 reviews "App crashes when uploading photos; whenever I try to upload my photos, an error occurs" and "App is crashing a lot recently". The feature volume for *upload photos* is 1, for only the first review mentions that feature.

Step 3: Generate RE-SWOT matrix. The FPS scores from Step 2 are used to generate the RE-SWOT matrix (illustrated earlier in Table 1). For each feature, the scores for each app are evaluated according to two criteria:

- *Positive/negative/neutral FPS.* A FPS is *positive* if Equation 2 holds true, *negative* if Equation 3 is true, and *neutral* when the FPS is within the range $(-\sigma, +\sigma)$. Based on the results of an exploratory study in which we applied our formulas to a few apps and their reviews, we pragmatically set σ to 0.1; in future work, more rigorous experimentation and tuning are necessary.

$$FPS_{i,j} \geq \sigma \tag{2}$$

$$FPS_{i,j} \leq -\sigma \tag{3}$$

- *Feature performance in the market.* We determine if a feature f_j is unique, above or below market average ($\overline{FPS_j}$). A FPS is *above average* if Equation 4 holds true, *below average* if Equation 5 applies. Moreover, a feature f_j is *unique* when only app a_i has reviews concerning f_j.

$$FPS_{i,j} - \overline{FPS_j} \geq \sigma \tag{4}$$

$$FPS_{i,j} - \overline{FPS_j} \leq -\sigma \tag{5}$$

Features from the competition with a positive FPS that is above the market averages are classified as *threats*. On the other hand, features with a negative FPS and below the market average represent *opportunities*. If the FPS refers to a feature of the reference app, it can be a *strength* (FPS is positive and above the market average) or a *weakness* (FPS is negative and below the market average). Feature that do not fit the aforementioned scenarios are not classified.

Table 2. The RE-SWOT matrix applied to photo editing apps.

Feature performance \ App with the feature	*Photo1*	*Photo2 or Photo3*
Positive and above market average	**Strengths:** filters	**Threats:** edit photos, syncing (Photo2) save photos (Photo3)
Negative and below market average	**Weaknesses:** save photos	**Opportunities:** filters, save photos (Photo2) exporting (Photo3)

Step 4: Generate requirements. In SWOT analysis, the TOWS framework [29] is used to identify strategies that can improve a company's competitiveness. In RE-SWOT, we adapt TOWS to identify the most suitable requirements for the app to excel in the market. With examples from the RE-SWOT matrix of Table 2, we illustrate the four types of requirements originating from TOWS:

- **SO requirements** aim at pursuing opportunities that fit well with the strengths. For example, feature *filters* should be boosted to exploit the opportunity that stems from the negative appreciation of filters in Photo2.
- **WO requirements** aim at overcoming weaknesses to pursue opportunities. For instance, the *save photos* weakness could be overcome by leveraging the opportunity given by the negative appreciation of that feature in Photo2.
- **ST requirements** aim at using strengths to reduce vulnerability to threats. No examples of this category exist in Table 2.
- **WT requirements** aim at minimizing weaknesses to make them less susceptible to threats. For example, Photo1 could imitate the implementation of the feature *save photos* in Photo3, which is currently a threat.

4 Prototype Tool

We implemented a tool that automatically creates an RE-SWOT matrix starting from a set of user reviews for the reference app and its competitors. The tool is built in R and Tableau Software and is available as an open source project[3].

The tool consists of two modules: (i) an *NLP module* implemented in R that creates the RE-SWOT matrix; and (ii) a *visualization module* for the analyst to interact with an RE-SWOT matrix, which is built using Tableau software. Both modules can be deployed through a Shiny web application.

4.1 NLP module

First, the module pre-processes the user reviews through the following steps:

[3] https://github.com/RELabUU/RE-SWOT

1. **Tokenization:** the reviews are split into sentences and words via *Udpipe*.
2. **To lowercase:** all tokens are converted to lowercase to make them uniform.
3. **Stopword removal:** common English words are removed using the stopword list of the *tm* package; moreover, additional words that are commonly found in reviews (e.g., the app name, "feature", "app") are filtered out.
4. **Noun, verb, and adjective extraction:** features are more likely to be described through nouns, verbs, and adjectives [12]. Thus, we used *Udpipe*'s POS tagging to select the tokens that meet those POS tags.
5. **Lemmatization:** we apply *Udpipe*'s lemmatizer to the tokens so that words such as "photos" and "photo" are reduced to the common term "photo".

In line with previous studies [12], we identify features through a collocation finding algorithm that identifies pairs of words (nouns, adjectives, verbs) that co-occur often in the reviews of each app. We exclude pairs that co-occur up to three times, and collocations that follow the patterns (adj, adj), (verb, adj), and (verb, verb), for our manual inspection revealed that they did not extract meaningful features. Hence, the considered collocation patterns are (noun, noun), (noun, adj), (noun, verb), (adj, noun), (adj, verb), and (verb, noun).

To further cope with the heterogeneous wording that users employ to refer to a same feature, we merge similar features (e.g., "photo edition" and "edit picture") by invoking the Cortical.IO service[4] to compute the cosine semantic similarity between all combinations of features. We merge feature labels with similarity score ≥ 0.60, and assign as label that having the highest frequency.

4.2 Visualization module

This module is an Information Visualization approach for analysts to interact with the RE-SWOT matrix. We describe it via Pfitzner's framework *et al.* [25].

Data factor. Three data objects are used: reviews, extracted features, and apps. Reviews have a date, a title, and a rating. A feature has a name, the related app, the FPS score, and the feature volume. An app can be classified as reference or competitor. Two relationship link the objects: (i) $mention(f, r)$ denotes that feature f is mentioned in review r, and (ii) $SWOT(f, a, c)$ denotes the class c (strength, weakness, opportunity, or threat) of feature f for app a.

Task factor. This focuses on what actions the user can perform on the data and is described according to Shneiderman mantra's dimensions overview, zoom, filter, and detail-on-demand [27]. An overview of the tool is shown in Fig. 2.

Overview. The user can see a set of circles, each representing a feature. A feature's x-axis position represents the app where the feature was identified, and the y-axis position represents the feature's uniqueness. The size of the circle corresponds to the feature frequency in the reviews, and its color illustrates its SWOT classification from the perspective of the reference app.

[4] http://www.cortical.io/compare-text.html

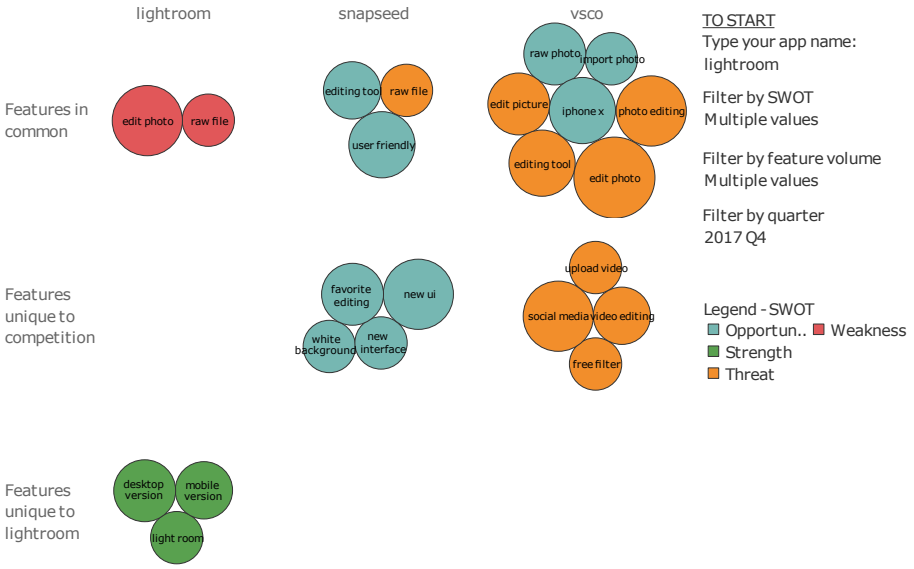

Fig. 2. The visualization of features in the RE-SWOT tool.

Zoom. The tool allows to zoom into a feature via a click, showing only the selected feature across all the apps with reviews mentioning that feature.

Filter. The tool allows to filter the visualized information in different ways:

1. By quarter. In Fig. 2, 2017 Q4 is shown;
2. By feature volume: *low* when the review volume is lower or equal to 1/6 of the range, *medium* if between 1/6 and 2/6 of the range, and *high* otherwise. In Fig. 2, only features with medium and high volume are shown;
3. By SWOT classification. In Fig. 2, features that are not assigned any of the four SWOT classes are omitted.

Details-on-demand. If necessary, the analyst can request details about (i) a feature, through a tooltip that shown the distribution of user ratings and other information concerning the feature (Fig. 3); and (ii) a review, showing the entire sentence in which a feature is mentioned.

5 Evaluation

We performed a preliminary evaluation aimed to determine how practitioners find RE-SWOT supportive to requirements elicitation through competitor analysis. In particular, we conducted three semi-structured interviews with three product management members from different app developers.

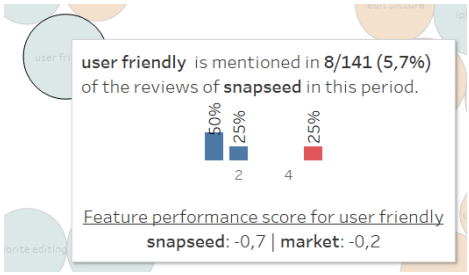

Fig. 3. Inspecting the details of a feature.

Interview protocol. Due to logistic constraints, the interviews were conducted remotely via online meetings that allowed for screen sharing and recording. The interview protocol, fully described in the appendix of [24], included five parts:

1. *Introduction* (5 min): an explanation of the research and the interview goals.
2. *Contextual questions* (10 min) were made concerning the company, the interviewee's role, the product, and their current app review analysis practices.
3. *Demo* (10 min): the tool was showcased on a set of photo editing apps.
4. *Tool use* (20 min): the participant could interact freely with the tool prepared with reviews concerning the interviewee company's app and its competitors. In this phase, the researcher minimized interference, while think aloud was encouraged, and the participant was allowed to stop before the time expired.
5. *Follow-up questions* (15 min) on the participant's experience with the tool, pros and cons, missing features, and a comparison with the current practice.

Two participants were recruited through a post in the online community MindTheProduct, an international community for product management. One participant was recruited via convenience sampling. All participants represented companies with a mobile app on Google Play or iOS App Store, and could identify 2+ competing app. Both the reference app and the competitors needed to have 200+ reviews in English over the same period and distribution platform.

Case descriptions. For confidentiality restrictions, we cannot disclose the identity of the companies. Pseudonymized data, including the periods from which we mined the reviews, is summarized in Table 3 and described below:

- *Case 1: dating apps.* We interviewed a senior business analyst, working for 2 years at a Canadian company whose service (*dat-ref*) has 150 million registered users; besides mobile apps for iOS, Android, and Windows Phone, *dat-ref* users can also use a website. As competitors, the interviewee suggested the market leader (*dat-c1*) and a fast-growing company (*dat-c2*).
- *Case 2: travel apps for tourists.* The reference app (*trv-ref*) supports the booking of activities and tickets, and is produced by a company with 400 employees distributed over three continents. We interviewed a senior product

manager with 6+ years of expertise in the field, who is responsible for the development roadmap. The competitors are an app for the Asian market (*trv-c1*) and the market leader (*trv-c2*).
– *Case 3: puzzle games.* The reference app (*pzl-ref*) has 135 puzzle types and is created by a small European company (6 employees). We interviewed the CEO of the company, who is a software engineer and has 9 years of experience in app development. The competitors *pzl-c1* and *pzl-c2* are similar apps suggested by the interviewee.

Table 3. Overview of the data collected for the three cases. The number of features refers to those that were *mined* by our tool.

Case 1: Dating apps May 2018			**Case 2: Travel apps** Dec 2016 – May 2018			**Case 3: Puzzle games** Jan 2017 – Jun 2018		
ID	**Reviews**	**Features**	**ID**	**Reviews**	**Features**	**ID**	**Reviews**	**Features**
dat-ref	3,220	280	trv-ref	253	9	pzl-ref	743	11
dat-c1	992	46	trv-c1	506	30	pzl-c1	2,105	72
dat-c2	802	66	trv-c2	375	14	pzl-c2	3,321	176

Results: current practice. All interviewees reported that they read the reviews for their app to some extent. The *dat-ref* interviewee reads all reviews approximately once per month to understand user perception and to identify areas for improvement. The reviews of *trv-ref* are automatically re-posted to the development team communication channel, giving everyone the opportunity to read them on a continuous basis. The CEO of *pzl-ref* reads the reviews occasionally through the Google Play Developers console, but found them only mildly useful for guiding product development. None of the interviewees has the habit of reading their competitors reviews, either due to time constraints (*dat-ref*) or because the option has never been considered (*trv-ref* and *pzl-ref*).

Positive aspects and insights. All participants referred to the visual and interactive aspect of RE-SWOT as the main positive feature of the tool. According to the *dat-ref* interviewee, "The tool is easy to learn and navigate", while the *trv-ref* interviewee described the experience as a "deep-dive analysis".

The *dat-ref* product manager highlighted other positive aspects: the possibility to zoom and see details can impact positively the communication between stakeholders, and the automated detection of phrases from the reviews is useful to work with a large volume of reviews. The same interviewee was surprised to discover that one of the competitors received fewer reviews than *dat-ref*.

The *trv-ref* interviewee stated to have become aware of a feature from the competitors that was previously unknown, and this—if time allows—may influence product development. It was also possible to see that one competitor was conducting a promotion because people are commenting on promo codes.

The CEO of *pzl-ref* said that the tool confirms that competitor apps adopt a similar business model, but found the insight generation limited, mostly due to the nature of reviews received by games, which are generic and not informative.

Improvements and missing features. Both the *dat-ref* and the *trv-ref* interviewees indicated the possibility to see trends over time as the most valuable improvement. While the *dat-ref* product manager found that the feature could make the tool directly usable in practice, the *trv-ref* interviewee would have liked to see the reviews for a custom period of time, instead of selecting one quarter.

The *trv-ref* participant observed some weaknesses in the feature extraction algorithm; despite our attempt to merge some features via lemmatization, there are still cases in which two word collocations are not merged automatically.

Concerning the SWOT classification, both the *trv-ref* and the *pzl-ref* participants found some opportunities to be inaccurate, for bugs in other apps do not necessarily represent an opportunity (*trv-ref*), and because some opportunities referred to features that are already implemented in the reference app (*pzl-ref*).

Finally, the *pzl-ref* interviewee suggested some usability improvements, including (i) a filter for the analyst to remove uninformative (e.g., short) reviews; and (ii) an automatically generated preview of the reviews mentioning a given feature, a sort of summarization.

Comparison to current practice. None of the participants currently read their competitors reviews; thus, the interviewees answered based on the way they read their own app's reviews. The *dat-ref* interviewee found the feature generation algorithm to be an improvement over their current practice. The *trv-ref* product manager thought that the visualization tool has the potential to deliver knowledge about the competitors. Finally, the *pzl-ref* interviewee referred us to the Google Play Console, which provides review highlights using similar techniques to RE-SWOT for feature extraction. However, according to the participant, analyzing reviews through RE-SWOT is more visual and interactive.

Factors influencing adoption. The *dat-ref* interviewee could see RE-SWOT being integrated into their workflow if more competitors could be handled (currently, it supports two competitors besides the reference app), and if trend analysis was included. As a drawback, the tool uses Tableau Software, which could make it unaffordable to some companies. Also the *trv-ref* participant thought they could adopt the tool, when a functionality was created to support the analysis of trends. On the other hand, the *pzl-ref* CEO found no real incentive to adopt the tool, unless a notification mechanism is put in place so that changes in sentiment are automatically pushed to the product management.

6 Discussion and Future Work

We have presented the RE-SWOT method for eliciting requirements for mobile apps based on competitor analysis through the automated processing of user re-

views posted in app stores. RE-SWOT draws inspiration from strategic planning and classifies app features as strengths, weaknesses, opportunities, and threats.

RE-SWOT employs NLP algorithms to automatically extract features from the reviews and to classify the features according to the SWOT framework. The results are rendered in an interactive visualization that helps analysts explore their app's market and identify possible requirements for the next releases.

Main findings. The *feature extraction* algorithm was evaluated positively by the *dat-ref* participant, while the other two interviewees found that the results include too many false positives. It is worth noting that the *dat-ref* receives 100 times the rate of reviews than the other reference apps; as such, we hypothesize that the algorithm is more effective for apps with a high review volume.

The *SWOT classification* was understood quite well by all interviewees. The most recurrent feedback was that not all strengths or weaknesses of the competitors represent actual threats or opportunities. Our to-be-expected conclusion is that the insights that RE-SWOT returns do not automatically result in new requirements for the reference app; human analytical skills are essential.

The *interactive visualization* of the tool was indicated as a positive factor by all interviewees. On the other hand, they indicated the need to include *change detection* techniques, either via trend analysis or through a notification system that informs analysts of significant changes in the sentiment toward a feature.

Validity evaluation. We discuss the major threats to validity based on the distinction between internal, conclusion, construct, and external validity [30].

Internal validity. All of our interviewees adopted practices in which competitor reviews were not considered. Therefore, the positive appreciation could be explained by the fact the interviewees had never analyzed competitor reviews. Moreover, the review volume differed greatly across apps, therefore resulting in a large variability in terms of the period we analyzed.

Conclusion validity. The interviews were manually coded into observations, and therefore we may have inserted our own beliefs during this process. Moreover, the number of cases ($N = 3$) is insufficient to draw definitive conclusions.

Construct validity. RE-SWOT has been evaluated as a whole rather than by its parts. Therefore, we cannot determine, for example, the extent to which the SWOT classification mechanism is effective, for the practitioners may have focused on other factors like the size of the circles (the volume).

External validity. As observational case studies were applied, there is no population to sample from. Therefore, generalization is analytical rather than statistical. Moreover, the number of reviews for case 2 and case 3 are rather low to justify an automated approach; as such, the case of dating apps is probably the most representative of the intended audience.

Future directions. Many improvements to RE-SWOT are possible. For feature extraction, we adapted Guzman and Maalej's technique [12], but we could experiment with the SAFE framework [17] or other recent algorithms. Also, the thresholds we used to determine the SWOT classes should be adjusted. Furthermore, as pointed out by the interviewees, trend analysis should be included for increasing the impact of the tool. The sentiment score calculation can be improved by looking at the text characteristics, instead of using only the number of stars assigned by the users. Inevitably, more empirical evaluation is necessary; in particular, we need to assess the effectiveness of the framework when used in the daily practice of requirements engineers and product managers.

In general, this work offers numerous opportunities for research that combines NLP and information visualization in RE; for another example of such synergy, see our work on terminological ambiguity [6]. In order for automated techniques to become useful for practitioners, the results of automation have to be turned into requirements analytics tools [26] that are built for use by human analysts.

References

1. Bano, M., Zowghi, D.: A systematic review on the relationship between user involvement and system success. Information and Software Technology **58**, 148–169 (2015)
2. Berry, D.: Natural language and requirements engineering—nu? In: International Workshop on Requirements Engineering (2001)
3. Bourque, P., Fairley, R.E.: Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press (2014)
4. Chen, N., Lin, J., Hoi, S.C.H., Xiao, X., Zhang, B.: AR-miner: mining informative reviews for developers from mobile app marketplace. In: Proc. of ICSE. pp. 767–778 (2014)
5. Cooper Jr, J.R., Lee, S.W., Gandhi, R.A., Gotel, O.: Requirements engineering visualization: A survey on the state-of-the-art. Proc. of REV pp. 46–55 (2009)
6. Dalpiaz, F., van der Schalk, I., Brinkkemper, S., Aydemir, F.B., Lucassen, G.: Detecting Terminological Ambiguity in User Stories: Tool and Experimentation. Information & Software Technology (2019)
7. Di Sorbo, A., Panichella, S., Alexandru, C.V., Shimagaki, J., Visaggio, C.A., Canfora, G., Gall, H.C.: What would users change in my app? Summarizing app reviews for recommending software changes. In: Proc. of FSE. pp. 499–510 (2016)
8. Fu, B., Lin, J., Li, L., Faloutsos, C., Hong, J., Sadeh, N.: Why people hate your app: Making sense of user feedback in a mobile app store. In: Proc. of SIGKDD. pp. 1276–1284 (2013)
9. Gao, C., Wang, B., He, P., Zhu, J., Zhou, Y., Lyu, M.R.: PAID: Prioritizing app issues for developers by tracking user reviews over versions. In: Proc. of ISSRE. pp. 35–45 (2016)
10. Genc-Nayebi, N., Abran, A.: A systematic literature review: Opinion mining studies from mobile app store user reviews. Journal of Systems and Software **125**, 207–219 (2017)
11. Groen, E.C., Seyff, N., Ali, R., Dalpiaz, F., Doerr, J., Guzman, E., Hosseini, M., Marco, J., Oriol, M., Perini, A., Stade, M.: The Crowd in Requirements Engineering: The Landscape and Challenges. IEEE Software **34**(2), 44–52 (2017)

12. Guzman, E., Maalej, W.: How do users like this feature? A fine grained sentiment analysis of app reviews. In: Proc. of RE. pp. 153–162 (2014)
13. Harman, M., Jia, Y., Zhang, Y.: App store mining and analysis: MSR for app stores. Proc. of MSR pp. 108–111 (2012)
14. Hill, T., Westbrook, R.: SWOT analysis: it's time for a product recall. Long range planning **30**(1), 46–52 (1997)
15. Hosseini, M., Phalp, K., Taylor, J., Ali, R.: Towards crowdsourcing for requirements engineering. In: Proc. of REFSQ. pp. 82–87 (2014)
16. Jin, J., Ji, P., Gu, R.: Identifying comparative customer requirements from product online reviews for competitor analysis. Engineering Applications of Artificial Intelligence **49**, 61–73 (2016)
17. Johann, T., Maalej, W.: Democratic mass participation of users in Requirements Engineering? In: Proc. of RE. pp. 256–261 (2015)
18. Kabbedijk, J., Brinkkemper, S., Jansen, S., van der Veldt, B.: Customer involvement in requirements management: lessons from mass market software development. In: Proc. of RE. pp. 281–286 (2009)
19. Keertipati, S., Savarimuthu, B.T.R., Licorish, S.A.: Approaches for prioritizing feature improvements extracted from app reviews. Proc. of EASE pp. 1–6 (2016)
20. Maalej, W., Nabil, H.: Bug report, feature request, or simply praise? On automatically classifying app reviews. In: Proc. of RE. pp. 116–125 (2015)
21. Maalej, W., Nayebi, M., Johann, T., Ruhe, G.: Towards Data-Driven Requirements Engineering. IEEE Software **33**, 1–6 (2015)
22. Pagano, D., Maalej, W.: User Feedback in the AppStore: An Empirical Study. Proc. of RE pp. 125–134 (2013)
23. Panichella, S., Di Sorbo, A., Guzman, E., Visaggio, C.A., Canfora, G., Gall, H.C.: How can I improve my app? Classifying user reviews for software maintenance and evolution. In: Proc. of ICSME 2015. pp. 281–290 (2015)
24. Parente, M.G.: Using NLP and Information Visualization to analyze app reviews. Master's thesis, Utrecht University, the Netherlands (2018), https://dspace.library.uu.nl/handle/1874/368082
25. Pfitzner, D., Hobbs, V., Powers, D.: A Unified Taxonomic Framework for Information Visualization. In: Proc. of APVis (2003)
26. Reddivari, S., Rad, S., Bhowmik, T., Cain, N., Niu, N.: Visual requirements analytics: A framework and case study. Requirements Engineering **19**(3), 257–279 (2014)
27. Shneiderman, B.: The eyes have it: A task by data type taxonomy for information visualizations. In: Proc. of VL/HCC. pp. 336–343 (1996)
28. Srivastava, P.K., Sharma, R.: Crowdsourcing to elicit requirements for MyERP application. In: Proc. of CrowdRE. pp. 31–35 (2015)
29. Weihrich, H.: The tows matrixa tool for situational analysis. Long range planning **15**(2), 54–66 (1982)
30. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in software engineering. Springer Science & Business Media (2012)
31. Yang, H., Liang, P.: Identification and Classification of Requirements from App User Reviews. In: Proc. of SEKE. pp. 7–12 (2015)