



Universiteit Utrecht

Utrecht University
Department of Information and Computing Sciences
Master in Business Informatics

**Master's thesis:
Using NLP and Information Visualization to
analyze app reviews**

Micaela Garcia Parente
m.garciaparente@students.uu.nl

1st supervisor: Fabiano Dalpiaz
2nd supervisor: Jan Martijn van der Werf

2018

Abstract

App store reviews are an abundant source for requirements elicitation, from which analysts can identify bugs, feature requests, and potential improvements for mobile apps from a large, distributed audience. As a paradigm shift occurs in the Requirements Engineering process, product development teams need new tools and techniques to support them in making real-time decisions based on user feedback data.

Natural language processing (NLP) techniques can contribute to this problem by offering automated means to do preprocessing, text classification, feature extraction, and topic modelling. On the other hand, Information Visualization can be applied to RE as a way to augment an analysts cognition and shorten the path from data to decision. When certain principles and guidelines are followed, an IV tool can speed up the knowledge discovery process, thus benefiting the analyst work.

In this context, we propose RE-SWOT, an approach and a tool that uses NLP and IV to support requirements elicitation from app store reviews through competitor analysis. The usefulness of RE-SWOT for practitioners is evaluated by observational case studies with 3 companies from different industries and contexts.

Keywords: app store reviews, CrowdRE, NLP, information visualization, natural language processing, requirements elicitation, requirements engineering

Acknowledgements

This thesis marks an important period of my trajectory that was both exciting and challenging. Completing a Master's course abroad is a journey on its own, and I would like to thank all the people that supported me on this process.

Thanks to Fabiano, for being the best supervisor I could ever have hoped for. This research would not be possible without all your constructive feedback, availability, and motivation.

For making the case studies possible, I would like to thank the participants from POF, GetYourGuide and Blob Connect. Your availability and feedback was fundamental in this research.

Thanks to Felipe, my partner in crime. For joining me in this rewarding adventure abroad and giving me love and laughs every single day. For holding the fort and helping me along this thesis, in all levels. You make me a happy person.

Finally, I would like to thank my family, for always supporting me. My parents, for all the love and for giving me the privilege of good education in such an unequal country. Alexandre, my second father, for inspiring me and sharing your knowledge. Vovó Ana, for being the kindest person I know. Muito obrigada!

Contents

Abstract	i
Acknowledgements	iii
1 Introduction	1
1.1 Problem statement	1
1.2 Research questions	2
1.3 Research method	2
1.3.1 Problem investigation	2
1.3.2 Treatment design	3
1.3.3 Treatment validation	3
2 Literature study	4
2.1 Literature study protocol	4
2.2 Crowd-based requirements engineering	5
2.2.1 What is Crowd-based Requirements Engineering	5
2.2.2 Requirements elicitation	6
2.2.3 Types of user feedback	6

2.2.4	App store reviews in requirements elicitation	9
2.3	Natural language processing	13
2.3.1	NLP techniques for RE	14
2.3.2	Opinion mining	16
2.3.3	Mining app store reviews for requirements elicitation	17
2.4	Information visualization	24
2.4.1	What is Information Visualization	24
2.4.2	Developing an IV tool	25
2.4.3	IV for requirements engineering	28
2.5	Literature findings	36
3	Tool development	38
3.1	Competitive analysis for requirements elicitation	38
3.1.1	Conceptual approach	38
3.1.2	Example	43
3.1.3	Tool architecture	46
3.2	Competitive analysis for requirements elicitation via NLP	50
3.2.1	Preprocessing	50
3.2.2	Feature extraction	51
3.2.3	Feature grouping	51
3.2.4	Feature tagging	51
3.3	Competitive analysis for requirements elicitation via IV	52

3.3.1	Data factor	52
3.3.2	Task factor	53
4	RE-SWOT evaluation	59
4.1	Evaluation protocol	59
4.1.1	Goal	59
4.1.2	Interview protocol	59
4.1.3	Sampling	60
4.2	Case descriptions	60
4.2.1	PlentyOfFish	60
4.2.2	GetYourGuide	61
4.2.3	Blob Connect	63
4.3	Results and conclusions	64
4.3.1	Current practice	64
4.3.2	Positive aspects and insights generated	64
4.3.3	Improvements and missing features	65
4.3.4	Comparison to current practice	66
4.3.5	Factors influencing adoption	67
4.3.6	Conclusions	68
5	Discussion	71
5.1	Answering the sub research questions	71
5.2	Answering the main research question	74

5.3	Limitations	75
5.3.1	Internal validity	76
5.3.2	Conclusion validity	76
5.3.3	Construct validity	76
5.3.4	External validity	77
5.4	Future work	77
	Appendix A Interview protocol	79
	Bibliography	82

List of Tables

2.1	Summary of classification-focused studies	20
2.2	Visualization categories [29]	31
3.1	Example review set	44
3.2	Illustrative example of step 4	47
3.3	Data types	53
3.4	Data relationships	53
4.1	Data collected for POF (period: from 01/05/2018 to 31/05/2018)	62
4.2	Data collected for GetYourGuide (period: from 01/12/2016 to 31/05/2018) . . .	62
4.3	Data collected for Blob Connect (period: from 01/01/2017 to 15/06/2018) . . .	63

List of Figures

2.1	Classification of user input [55]	8
2.2	Example of user review of Facebook on Google Play Store	8
2.3	Example of user review of Facebook on iOS App Store	8
2.4	Appbot	12
2.5	Appfollow	12
2.6	App-specific features analysis	13
2.7	Group ranking equations [27]	22
2.8	AR-Miner summary [27]	22
2.9	Product review summary [47]	23
2.10	Di Sorbo et al. summary [33]	23
2.11	Example of preattentive task: detecting the presence of a red circle	26
2.12	IV framework [66]	27
2.13	Requirements analytics framework [70]	29
2.14	RE lifecycle [29]	30
2.15	Appfollow visualization - Rating over time [16]	33
2.16	Appbot visualization - Versions over time [1]	34

2.17	Appbot visualization - Word cloud of popular words [1]	34
2.18	Appbot visualization - Reviews per emotion [1]	34
2.19	Conceptual goals and their operational questions to be addressed by a visual requirements analytics approach [70]	34
2.20	Evaluation for Appflow visualization - Rating over time	35
2.21	Evaluation for Appbot visualization - Versions over time	35
2.22	Evaluation for Appbot visualization - Wordcloud	35
2.23	Evaluation for Appbot visualization - Reviews per emotion	35
3.1	SWOT matrix	39
3.2	SWOT analysis adapted to Crowd-based Requirements Engineering	40
3.3	RE-SWOT method	41
3.4	Illustrative example of step 2	45
3.5	Illustrative example of step 3	45
3.6	Tool architecture - Context viewpoint	47
3.7	Tool architecture - High-level functional viewpoint	48
3.8	Tool architecture - Functional viewpoint of the NLP module	49
3.9	Tool architecture - Functional viewpoint of the visualization module	49
3.10	Overview of features	54
3.11	Zoom into a feature	55
3.12	Filter by quarter	55
3.13	Filter by feature frequency	56
3.14	Feature frequency categories	56

3.15	Filter by SWOT classification	56
3.16	Details about a feature	57
3.17	Details about a review	57
3.18	Reviews mentioning a feature	57
3.19	History options	58
3.20	Exclude a feature	58
4.1	POF app on Google Play [7]	61
4.2	GetYourGuide app on Google Play [5]	62
4.3	Blob Connect app on Google Play [3]	63
4.4	Review highlights from Google Play Developers Console[12]	67

Chapter 1

Introduction

1.1 Problem statement

User feedback is considered as a central piece of requirements gathering. While much of this feedback is provided through organized sessions with user representatives, crowd-based channels like social media and online forums are an increasingly common way for users to express their opinions about a software product.

However, analyzing crowd-based information requires different approaches than traditional Requirements Engineering. In this context, Crowd-based requirements Engineering (CrowdRE) concerns automated or semi-automated methods to gather and analyze information from a crowd, ultimately resulting in validated user requirements [42].

When analyzing unstructured textual reviews, Natural Language Processing (NLP) techniques have been employed towards goals such as summarizing and classifying user input into structured knowledge to help practitioners in requirements gathering. As an example, Guzman and Maalej [43] proposed an automated approach for fine grained sentiment analysis of app reviews, which enables extracting app features and their sentiment in the reviews.

Besides relying on NLP techniques, the proposed thesis investigates Information Visualization as a way to help practitioners analyse their mobile apps user reviews. In particular, the project aims at developing a visualization tool to help product managers and requirements engineers identify new requirements from navigating information contained in the reviews.

As Cooper et al. [29] reported, research in the area of visualization concerning the early activities

of RE has received relatively little formal attention until recent years. According to the authors, much opportunity exists to develop visualizations that assist with the pre-requirements phases of understanding the context, which reinforces the relevance of the proposed research. The next sections elaborate on the research questions to be answered and the proposed methodology to tackle them.

1.2 Research questions

In this research, we aim to answer the following research question, which is decomposed into six sub-research questions:

MRQ: *How to aid requirements engineers analyse app reviews through NLP and Information Visualization?*

RQ1: *How do practitioners analyse app reviews?*

RQ2: *What obstacles do practitioners face when analysing app reviews?*

RQ3: *Which applications of NLP already exist in Crowd-based requirements engineering and what can be learnt from them?*

RQ4: *What types of Requirements Engineering visualizations already exist and what can be learnt from them?*

RQ5: *How to build a tool that uses NLP and IV to help requirements engineers analyse app reviews?*

RQ6: *Does the proposed tool satisfy the expectations of practitioners to help identify and prioritize requirements derived from app reviews?*

1.3 Research method

The design science methodology [84] provides a suitable framework for addressing the proposed research. In a design science project, researchers iterate over a design cycle composed of three phases: (i) Problem investigation, (ii) Treatment design, and (iii) Treatment validation.

1.3.1 Problem investigation

As a preparation for the artifact design, investigating the problem at hand is required. A literature review on Crowd-based requirements Engineering is performed, so to understand

current practices and challenges, and answer RQ1 and RQ2. Furthermore, the literature on NLP and IV is reviewed, by focusing on how these knowledge areas can contribute to the problem. As a result, sub-research questions RQ3 and RQ4 are also answered, respectively.

1.3.2 Treatment design

The treatment design is expected to answer RQ5. In the proposed research, the artifact under design is a visualization tool for crowd-sourced user reviews related to a mobile app. The tool is intended to help software product managers and requirements engineers to identify and prioritize new requirements derived from textual feedback submitted by app users.

The problem investigation described in section 1.3.1 will clarify what the current practice is and which criteria the artifact to be designed should satisfy to help requirement engineers analyze the reviews.

1.3.3 Treatment validation

The last phase of the design cycle is the treatment validation, which aims to answer RQ6. To understand how the developed tool can be useful to practitioners when analyzing app reviews, observational case studies are conducted with three commercial mobile applications: Plenty-OfFish, GetYourGuide, and Blob Connect.

The case studies use real review data collected by the research for each app and two competitors named by the interviewees. After participants use the tool, their feedback is collected about which aspects are the most useful to support requirements elicitation, and what could be improved to satisfy their expectations.

Chapter 2

Literature study

2.1 Literature study protocol

To address the research questions stated in chapter 1, the following areas were selected for the literature study: (i) Crowd-based requirements engineering, (ii) Natural language processing, and (iii) Information visualization. Rather than doing a comprehensive, systematic review of these topics, we gave focus to what can be learned about integrating crowdsourced feedback, more specifically app store reviews, into the requirements elicitation activity.

In particular, the following keywords were used as a starting point for each topic:

(i) Crowd-based requirements engineering: *“crowd-based requirements engineering”, “app review analysis”, “requirements analytics”, “requirements app reviews”*.

(ii) Natural language processing: *“app reviews nlp”, “app reviews natural language processing”, “crowd-based requirements engineering nlp”, “crowd-based requirements engineering natural language processing”*

(iii) Information visualization: *“information visualization”, “requirements engineering visualization”, “visualization tools requirements”*

The selection criteria for the literature was the following: the studies had to explicitly provide an understanding of one of the above fields, or demonstrate possible approaches that can be used to incorporate crowdsourced feedback (such as app store reviews) into requirements elicitation.

Based on the initial results obtained, the snowballing method was used to select new studies for the literature review. This way, we reduce the risk that relevant publications are left out

of the analysis. The process was stopped when no further relevant papers published after the year of 2000 could be found.

2.2 Crowd-based requirements engineering

The goal of this section is to build an understanding of the problem domain. We first investigate the new paradigm of crowd-based requirements engineering and how it impacts the requirements elicitation activity. Then, we give a brief overview of types of user feedback, and app store reviews characteristics. Finally, we analyze the role of app store reviews on requirements elicitation as reported on literature and practice.

2.2.1 What is Crowd-based Requirements Engineering

Groen et al. [42] defined Crowd-based Requirements Engineering (CrowdRE) as *“an umbrella term for automated or semiautomated approaches to gather and analyze information from a crowd to derive validated user requirements”*. Similarly, Hosseini et al. [46] discussed the concept of “feedback-based requirements engineering”, where users feedback (either implicit or explicit) is used by developers to understand the requirements of the next release of a software product. Although there is an overlap between the two definitions, “crowd-based requirements engineering” is more specific as it emphasizes that feedback is collected from the crowd, as opposed to small groups or user representatives.

This new paradigm has a direct impact on how requirements elicitation is performed by requirements engineers, analysts, developers, product managers, in short any stakeholder involved in software product evolution. As Groen et al. [42] discuss, although RE approaches to handle large amounts of feedback exist since the 2000s, the recent rise of platforms such as social media has significantly increased the volume of feedback given by the crowd.

Therefore, CrowdRE has become an emerging research field with most of its literature dating to the 2010s. With the ease of access to user feedback and usage data, and the pro-active participation of users online, authors foresee a shift towards data-driven user-centered development, prioritization, planning, and management of requirements [57].

Crowdsourcing is considered a particularly promising paradigm when eliciting requirements for highly interactive systems which are potentially used by a wide variety of users in a dynamic, perhaps unknown, context [46]. In such cases, changes to objectives, priorities and constraints happen constantly, triggering the need for real-time decisions and incremental adjustments [57].

As a consequence, new infrastructure, methodologies, and tools will become more and more important to support the development of products relying on deep customer insight and real-time feedback.

2.2.2 Requirements elicitation

In this section, we give a brief overview of what is requirements elicitation and what are the goals of requirements engineers during this step.

According to SWEBOK guide [22], *“Requirements elicitation is concerned with where software requirements come from and how the software engineer can collect them. It is the first stage in building an understanding of the problem the software is required to solve. It is fundamentally a human activity and is where the stakeholders are identified and relationships established between the development team and the customer.”* As the definition suggests, requirements elicitation is traditionally considered a design-time activity.

Upon identifying the relevant sources of requirements (such as stakeholders, market landscape, or business goals), the requirements engineer needs to elicit information from them, from which he/she formulates new requirements [22]. Eliciting requirements from human stakeholders is considered to be particularly challenging [22], and a number of techniques (such as interviews and prototypes) are available with this purpose.

Entering the CrowdRE perspective, Hosseini et al [46] extend the definition of requirements elicitation to not only include the identification of new requirements, but also the evaluation of existing software and the consideration of alternative solutions to meet users needs. This broader view on requirements elicitation reflects the change of paradigm discussed in Section 2.2.1, and builds upon new approaches [14] that advocate that this activity can also be done at runtime, rather than only design time.

As a consequence, even more emphasis is put on the importance of gathering and analyzing user feedback to identify new requirements and judge existing software implementations. In the next section, we discuss types of feedback and how they are used for eliciting requirements.

2.2.3 Types of user feedback

User feedback may be classified into different types in the literature. These categorizations are important to researchers and practitioners as different types of feedback might be associated with specific techniques and challenges.

Implicit vs. explicit feedback

A popular distinction [57] [46] [50] is made between implicit and explicit feedback. Implicit feedback is the feedback indirectly collected through the monitoring of patterns of use of the software. Data collected might include clicks and interactions on the user interface, which can be associated to specific app features, components and/or requirements. In the app development domain, numerous commercial tools (such as App Annie [15], AppAnalytics [2], or Mixpanel [61]) already exist and are widely used by teams to collect and analyze implicit feedback about their apps.

On the other hand, explicit feedback concerns opinions directly given by users about a software product. In a crowd-based scenario, explicit feedback is typically given in natural language through social media, online forums, or app stores. As opposed to implicit feedback, explicit feedback typically consists of unstructured data of varying quality. These characteristics make the analysis of explicit feedback less trivial, and therefore an active research topic: according to Daneva [32], 93% of the literature on crowd-sourced feedback studies its explicit form.

Push vs. pull feedback

Another classification found in the literature relates to the communication direction of feedback: a distinction is made between push feedback and pull feedback. As Groen et al. [42] discussed, push feedback happens when the crowd initiates the feedback, whereas pull feedback is given when communication is first initiated by the development team.

In crowd-based requirements engineering context, it is often the case that users are highly distributed and actively push a large number of issues, wishes and opinions to engineers [55]. As a result, navigating this large number of unsolicited feedback requires different practices than found in traditional requirements engineering.

Figure 2.1 combines these two classifications into quadrants and provides examples of feedback gathering techniques for each of them. Given these two dimensions, we introduce app store reviews as an instance of explicit, push feedback.

App store reviews

According to Jansen and Bloemendal [48], an app store is “*An online curated marketplace that allows developers to sell and distribute their products to actors within one or more multisided software platform ecosystems.*”. As the authors report, the most renowned app stores come

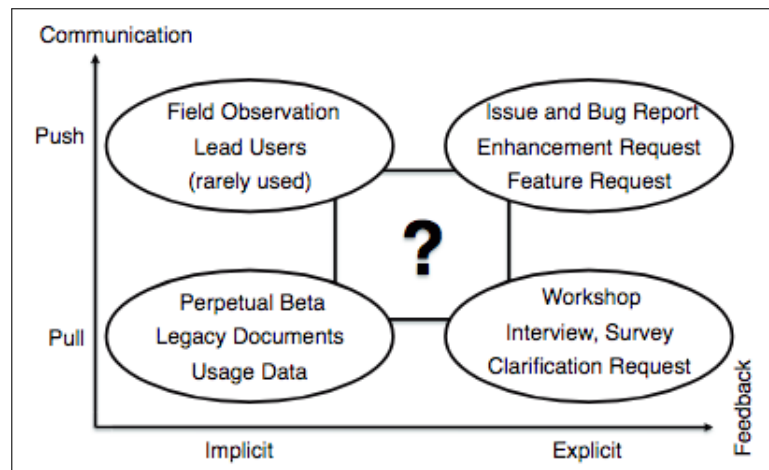


Figure 2.1: Classification of user input [55]

from mobile phone platforms, such as Android’s Google Play and Apple iOS App Store.

One of app stores core features is the ability for users to read and write reviews of apps. Figures 2.2 and 2.3 show examples of app store reviews of Facebook found on Google Play and iOS App store.

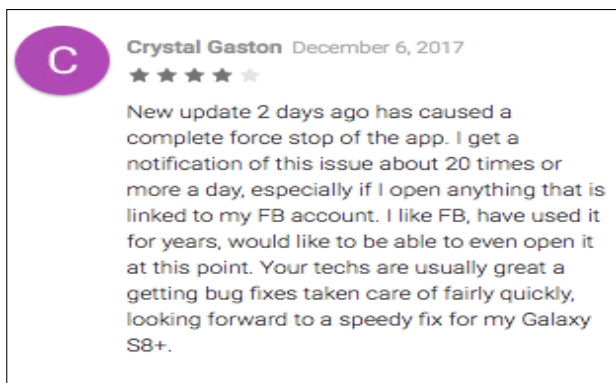


Figure 2.2: Example of user review of Facebook on Google Play Store

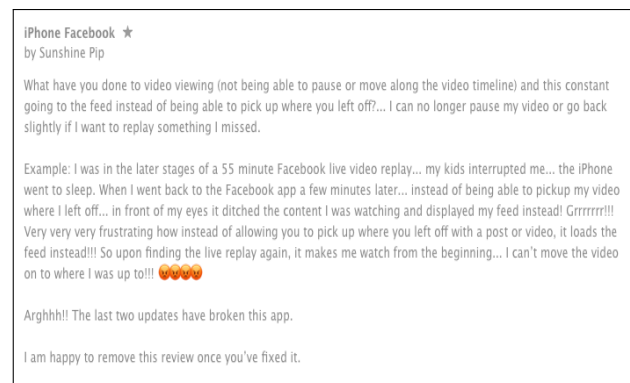


Figure 2.3: Example of user review of Facebook on iOS App Store

As it can be seen from the Figures, the reviews also contain metadata which varies according to the platform but can include a time stamp, app version rated, a numeric rating (“stars”) defined by the reviewer, a review title, and more. This richness of data offers many possibilities for practitioners and researchers on analysing feedback about existing apps.

Genc-Nayebi and Abran [40] did a systematic literature review on opinion mining from mobile app store reviews and found 24 relevant studies in the field. According to the authors, most studies obtained the data by using the search API and the RSS feed generator from Apple, or by using some type of scraping script. In the next section, we analyse the use of app store reviews for requirements elicitation.

2.2.4 App store reviews in requirements elicitation

App store reviews can be used both to evaluate current implementations of an app and to identify new requirements. Although a portion of reviewers simply describe their sentiment (e.g., “This is the best app ever!”, or “Terrible, don’t download it!”), users often mention aspects of the app in their reviews, which makes them especially insightful for requirements engineers.

Such aspects can be functional (e.g., “I used to love it, but I can’t watch videos anymore!”) or non-functional (e.g., whether the app is easy to use, or if there is a performance problem). Moreover, users might also comment on aspects that are not yet implemented, i.e., requests for improvements or new features.

Authors agree on the numerous advantages that app reviews have over other types of feedback. Potential benefits include better understanding how apps are actually used, quickly detecting bugs introduced by new releases [64], improving cost-efficiency, and getting insights from a wider, more diverse range of users [46]. On the other hand, some characteristics of app store reviews impose limitations for researchers and practitioners who are interested in analyzing them.

Challenges

Volume A study conducted in 2013 [64] showed that popular apps, such as Facebook, can get more than four thousand reviews in a single day. While such a large number of feedback contains much information that can be used to derive new requirements, processing and analyzing this volume of reviews requires more convenient solutions than a fully-manual approach.

On the literature, CrowdRe studies mainly rely on linguistic analysis techniques to automate this process [42] and improve efficiency over manual approaches. On the other hand, the definition of requirements elicitation as a “fundamentally human activity” [22] should not be overlooked. Therefore, new support tools must achieve a balance between automation and cognitive aspects of requirements elicitation [59], by “bringing the human into the loop and promoting thinking about the results” [20].

Noise As the result of a crowd-sourced mechanism initiated by final users, reviews are extremely noisy. A study conducted by Chen et al [27] revealed that only 35.1% of app reviews contain information that can directly help developers improve their apps. This makes system-

atic filtering and aggregation of content a fundamental step of any analysis [64].

Although there is no definitive agreement on what characterizes an informative review, authors contributed from different perspectives to this problem. Approaches studied include spam detection [26], “informative” review classification [27], and using taxonomies to classify and obtain insights about which topics are most relevant to developers [64] [65] [56]. For instance, Maalej et al [56] suggested that reviews containing bug reports or feature requests have a more immediate impact on RE process, as opposed to reviews consisting of simple praise or user experience narrations. Section 2.3 elaborates on the different NLP techniques applied in the literature to classify app reviews.

In summary, no one-fits-all approach has been found, suggesting the need to merge multiple criteria to allow analysts to filter and navigate the most useful reviews according to their specific needs.

Lack of context One of the drawbacks of app store reviews is that they often lack metadata about users and app usage. This has a negative impact when analysts try to assess issues reported, since users need to describe the sequence of interactions needed to reproduce the problem, and there is no consensus of what information must be included [55]. Thus, further iterations of elicitation are often needed to clarify the conditions where bugs appear.

Secondly, being unable to identify information such as age and gender from the reviewers makes it harder to create subgroups [42]. Analysts might want to know which types of users request a certain feature, but the lack of contextual information requires further investigation from the team.

One way to overcome this challenge is to embed interfaces for the submission of feedback into applications, which allows for the automatic transmission of contextual information [55]. There are numerous commercial tools for this purpose, such as Usabilla [81], UserVoice [9], and Apptentive [6], which typically employ a pull-based mechanism by prompting users to rate their experience or a particular feature during use.

Practitioners argue that such in-app feedback tools improve requirements elicitation as it establishes a private, two-way conversation between users and app developers. On the other hand, app store reviews are considered an one-way channel, where more extreme opinions can be found [17]. Rather than diminishing the importance of app store reviews, this perspective suggests that app store reviews can be especially important for finding critical issues that are

not necessarily reported in other channels. Besides that, as app store reviews can be read by other users and competitors, they can directly influence an app's reputation.

Conflicting opinions Finally, various authors [42] [76] [46] [57] [50] [51] cited the fact that reviews often contain conflicting opinions. Conflicts are a known topic in requirements elicitation literature: it is the role of the analyst to mediate conflicting opinions and find a suitable resolution through negotiation and compromise [87].

However, in the case of app store reviews, negotiating with users is often not straightforward, and analysts must use available information to base their decisions. As Keertipati et al [51] suggest, prioritization approaches can be used to weight issues found in the reviews according to their dimensions, such as their prevalence amongst the user population, and their seriousness. Different aspects, such as rating, sentiment, and frequency can be used to model these dimensions.

State of the practice

Generally speaking, limited literature is available about the use of app store reviews for requirements elicitation in real-world practice. Therefore, we additionally consider “grey literature” such as industry reports, blog posts, and informal input collected from mobile app product managers in specialized forums. Although the information provided by these sources lacks scientific backing and must be interpreted cautiously, it is a complement to the limited literature existent on the topic.

According to Keertipati [51], app development companies often employ so-called community managers, who are responsible for manually reading and replying to user reviews. During this process, new issues are eventually identified and raised to the team, leading to new requirements. Although a manual approach might suffice in some scenarios, it is time consuming, difficult to scale, and might suffer from bias [51] [42]. The experience shared by practitioners appears to be consistent with the literature: when asked about their preferred approaches or tools for analyzing app store reviews, the participants reported to manually read the feedback in order to identify potential improvements. Tools are reportedly used to facilitate parts of the process and reduce the workload (e.g., by filtering the low-rating reviews, adding tags to them, and/or connecting the reviews source to team communication channels).

In summary, when it comes to analyzing app store reviews, CrowdRE is not well established in the industry. Authors suggest that one of the reasons for this is that current platforms do

not allow practitioners to systematically filter, aggregate, and classify user feedback [64] [57].

When investigating available tools, two major players are found: Appbot [1] and Appfollow [16], illustrated by Figures 2.4 and 2.5.

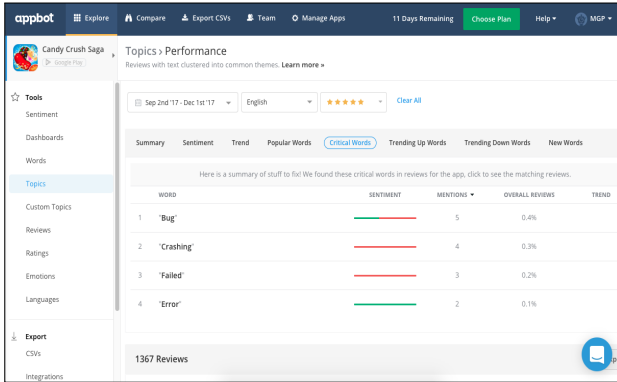


Figure 2.4: Appbot

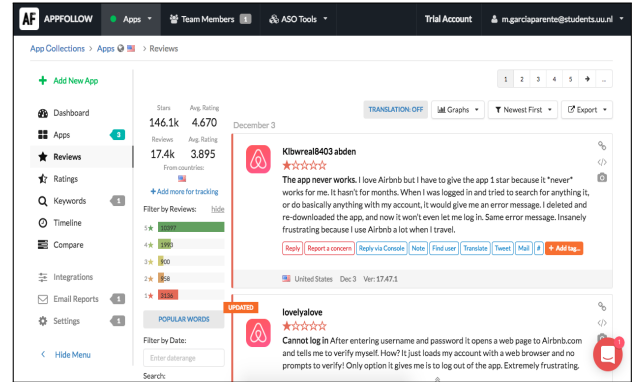


Figure 2.5: Appfollow

Upon selecting one or more mobile apps, Appfollow allows users to see the average rating and the distribution of stars over time, the main keywords found in the reviews (and their associated ratings), and offers the possibility to manually tag reviews. On the other hand, Appbot’s feature set is more rich in terms of linguistic analysis. The app tags the reviews according to 43 diversified predefined topics, ranging from operational system versions (such as “Android Kitkat” and “iOS 7”) to generic features (“Sign Up & Login”, “Payment”) and review purpose (such as “Bug” and “Feature request”). Besides that, it is possible to see the historical sentiment breakdown of the reviews, and their emotional dimensions (in scales ranging from “Assertive” to “Passive” and from “Pleased” to “Displeased”). Finally, both tools enable integration with 3rd-party tools for notifications and creation of customer service tickets.

In a blog post on a expert portal, Spiewanowski [75] argues that existing commercial tools employ simplistic approaches to analyze app store reviews. He suggests that analyzing the sentiment of generic topics does not suffice practitioners’ needs, and proposes an empiric methodology to obtain actionable feedback related to app-specific features:

“I suggest creating a spreadsheet in which to keep score. The researcher should give a positive score +1 for each positive comment on the feature, -1 for each negative, and 0 for neutral mentions. If the feature is not mentioned, the field should be left blank. With the spreadsheet ready the results can be easily aggregated to show the sentiment change between app versions and across time.”

On top of that, the author suggests the use of visualization as a way to obtain action points from the analysis: a matrix positioning app features according to their sentiment and frequency in the reviews is proposed, as Figure 2.6 shows. While the author highlights the benefits of the approach through a case study, he recognizes its limited scalability and calls for the development of automated tools that provide feature-specific summarization.

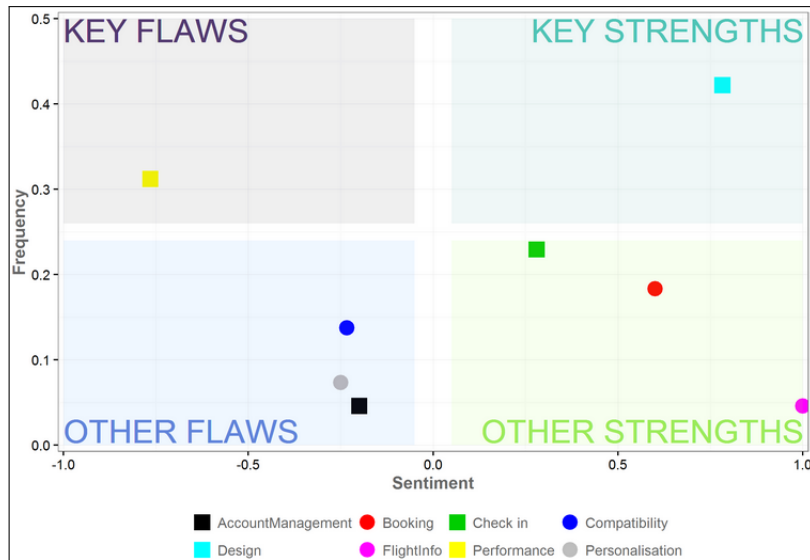


Figure 2.6: App-specific features analysis

This section has attempted to provide an overall understanding of the problem of integrating app store reviews into requirements engineering, both from academic and industrial perspectives. Eliciting requirements using app reviews is a non-trivial task that requires careful consideration of different aspects, such as features impacted, severity of issues, app versions affected, and so on. As the literature study showed, limited tools are available for commercial use and practitioners often use simplistic, sometimes manual approaches to accomplish their goals.

Next, we investigate how Natural Language Processing and Information Visualization can contribute to provide solutions to this problem. We analyze which applications exist, and which opportunities remain open to be addressed.

2.3 Natural language processing

In this section, we introduce NLP as a set of techniques that can support RE activities. Next, we investigate the field on opinion mining and focus on how it can be applied to app store reviews as a way to support the requirements elicitation activity.

2.3.1 NLP techniques for RE

According to Chowdhury [28], “*Natural Language Processing (NLP) is an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to do useful things*”.

NLP is a promising approach to support Requirements Engineering activities, since systems are becoming increasingly complex and difficult to manage from the RE perspective [34]. In this context, Falessi et al. classified several NLP techniques according to four dimensions: algebraic models, term extraction, weighting schema, and similarity metric. Although the authors focus on the problem of identifying similar requirements, their taxonomy gives insights about which NLP techniques can be used in the RE domain.

Algebraic models. Algebraic models evaluate the semantic similarity among words. Three techniques are characterized in this dimension: Vector Space Model (VSM), Thesaurus-based model, and Latent semantic analysis (LSA) [34].

- **Vector Space Model (VSM):** In this approach, text documents are represented as a vector in a multidimensional space where each dimension corresponds to a term. Two documents can be compared against each other by calculating the distance between their vectors. For instance, the sentences “Orange is a fruit” and “Orange is my favorite fruit” can be represented as the vectors $[1, 0, 1, 1, 0, 1]$ and $[0, 1, 1, 1, 1, 1]$ respectively, where the dimensions are $\{a, favorite, fruit, is, my, orange\}$.
- **Thesaurus-based:** In this approach, a thesaurus (such as WordNet [60]) with a given set of synonyms is used to compare terms.
- **Latent semantic analysis (LSA):** LSA represents an evolution to VSM in the sense that it considers synonyms, by looking at words that co-occur often in some text fragment. Given an input corpus, it outputs a thesaurus with the similarity among words.

Term extraction. Term extraction is a way to preprocess the text so that only words that carry meaning to the analysis are kept. The authors [34] distinguish between simple term extraction and POS tagging:

- **Simple term extraction:** This approach relies on tokenization and stopword removal. Tokenization transforms the text into a series of tokens where capitals, punctuation, and

brackets are removed. Stopword removal consists of removing words that do not convey meaning (based on a predefined stopword list), such as articles, pronouns, etc.

- **POS tagging:** This approach relies on tagging the role that each word plays in the text, such as noun, verb, or adjective. After the tokens are tagged according to their part-of-speech, they can be weighted by how much meaning their POS offers to the text [34].

Term weighting. Term weighting is a way to convey the importance of a term across documents. There are different ways to weight terms based on their occurrence in a text:

- **Raw frequency:** The number of times that the term occurs in the text.
- **Binary:** Assigns 1 if the term occurs in the text, and 0 if not.
- **Term Frequency (TF):** Also called Normalized Term Frequency, it represents the number of occurrences of a term in a document, divided by the length of the document, so to prevent that long documents generate frequent terms with little meaning. For example, the TF of “I” in the texts “I love you.” and “I think this product was well worth my money, I love it and I think everyone should buy it.” would be $1/3 = 0.33$ and $3/19 = 0.16$, respectively.
- **Inverse Document Frequency (IDF):** The log of the inverted fraction of documents that contain a term in a collection of documents. The idea behind this approach is that terms that consistently appear in many documents of a domain (e.g., “car” in automotive documents) should not receive a high weight.
- **TF_IDF:** The multiplication of TF and IDF. TF_IDF combines the benefits of TF and IDF, by considering the term frequency, but punishing terms that appear in virtually all documents.

Similarity metrics. There are different ways to compute the similarity level of two text fragments given the similarity levels of their terms.

- **Vector similarity metrics:** Calculates the similarity between two text fragments by measuring how similar their vectors are. Popular vector similarity metrics are Dice, Jaccard, and Cosine similarity [34].
- **WordNet similarity metrics:** Metrics offered by the thesaurus Wordnet. These are unidirectional, i.e., the similarity between “Manuscript” and “Document” is different

depending on its direction, since “Manuscript” can be replaced by “Document”, but the opposite is not always true [34].

2.3.2 Opinion mining

The area of NLP concerned with tracking what a public thinks about some product or topic is called opinion mining, or sentiment analysis. An opinion can be formally defined as a quintuple consisting of an entity, an aspect of such entity, a sentiment about the aspect, an opinion holder, and the time when the opinion was expressed [54]. For example, in the sentence *“I love this phone’s battery!”*, the first three components can be determined: battery is an aspect of entity mobile phone and a positive sentiment is expressed [79].

Based on this definition, opinion mining systems typically aim at identifying one or more of these components out of a text. Opinion mining can be made at three levels: the document level, the sentence level, or the aspect level [79].

At the document level, the goal is to obtain a sentiment orientation for the whole document, such a tweet, an Amazon review, or an article. To do so, classifiers such as Naive Bayes, Maximum Entropy Modeling (MaxEnt) and Support Vector Machines (SVM) are commonly used in the literature through a supervised approach. However, the need for an annotated corpora is a challenge, and the success of classifiers will often be tied to a specific domain, requiring that new models are trained for different areas [79]. Similarly, sentiment analysis at the sentence level is close to the previous case in the sense that sentences can be regarded as short documents.

On the other hand, aspect-level opinion mining aims to identify the sentiment towards a specific aspect (also known as a feature or attribute). Therefore, two sub-tasks are involved: first, the target aspects need to be extracted, and second the sentiment towards them must be classified. In the example *“I love this phone’s battery!”*, the outcome would be that a positive sentiment is expressed about the phone’s battery. In many cases, users have a positive sentiment about a feature but dislike another aspect of the product, thus a fine-grained analysis is justified.

However, doing opinion mining at this level of detail is typically even more challenging than at the document level, since the need for annotated corpora is more complex and requires more effort. Therefore, to perform such fine-grained analysis, researchers have started to apply unsupervised or semi-supervised approaches, such as Latent Dirichlet Allocation (LDA) and its variants [79].

2.3.3 Mining app store reviews for requirements elicitation

Opinion mining approaches are very much applicable to the app development domain. However, mining app store reviews can bring extra challenges, as pointed by Genc-Nayebi and Abran [40]. The granularity of aspects mentioned in the reviews tends to be higher, since users will not only comment on app features, but also specific parts of the interface, or particular interactions they had. Another difference with traditional reviews repositories is the text length: app store reviews are 71 characters on average, which is relatively short.

Despite these difficulties, the application of opinion mining to help developers elicit requirements is rising as an active research field. However, as reported by Genc-Nayebi and Abran [40], most works published are still exploratory, which leaves room for further progress. We analyze what can be learnt from these studies according to four perspectives: (i) Preprocessing reviews, (ii) Classifying review content, (iii) Extracting features, and (iv) Summarizing reviews.

Preprocessing reviews

One of the main challenges identified in Section 2.2.4 is that reviews are intrinsically noisy. While authors [26] [27] proposed ways to systematically filter out spam and non-informative reviews, most studies apply preprocessing tactics to reduce noise in the reviews. The most popular techniques are stopword removal, stemming and lemmatization, but these approaches need to be considered carefully depending on the modelling goal.

As Maalej and Nabil [56] discussed, removing common English words tends to improve classification accuracy. However, one must be careful since some words can be especially useful when predicting user intention. For instance, “should”, “must” might indicate a feature request, while “but”, “before”, “now” point towards bug reports.

Both lemmatization and stemming are used as alternatives to standardize different words with similar semantic meaning. While stemming reduces words to their root (“notifications” becomes “notif”), lemmatization only removes the words inflectional ends (“notifications” becomes “notification”). Although both approaches were found in the literature, authors point to the drawbacks of stemming. Gao et al. [39] argue that stemming is not suitable for reviews where there is a large number of casual words. Moreover, Maalej [56] highlights the fact that this technique will not consider parts-of-speech when standardizing words (e.g., the noun “goods” and the adjective “good” will both be reduced to “good”, even though they have different meanings).

Other approaches found include removing reviews with less than a certain number of words [86] [33], matching synonyms, and filtering words from specific POS tags (e.g., nouns, adjectives and verbs) [56]. Besides that, authors argue that sentence tokenization is an important preprocessing step, since a review might contain both informative and uninformative parts.

Classifying review content

Several studies focus on the classification of app store reviews according to a given taxonomy, by using a supervised approach. Authors used text from titles and reviews (by models consisting of their words, and/or their semantics), as well as metadata such as sentiment score, rating, and length to classify the reviews. The target categories are often predefined according to a goal derived from the requirements elicitation process.

For instance, Yang and Liang [86] worked on classifying reviews as functional requirements (FRs) or non-functional requirements (NFRs). Their approach consisted of using regular expressions to detect specific keywords that the authors found to be frequently associated with FRs or NFRs.

However, as Di Sorbo et al. [33] discuss, most classifications attempts in the literature focus on two aspects:

- **User intention:** represents user's goals when writing the review, e.g., reporting a bug, requesting a feature, expressing satisfaction, etc.
- **Review topic:** specific subjects being reviewed by the users, for example the app as a whole, its interface, or a particular feature.

Classifying app store reviews according to user intention and/or review topic is a way to tag the reviews for analysis. Such annotations are reportedly a way to help requirements engineers obtain a high-level understanding of the feedback provided (and optionally inspect reviews for more detail when needed), instead of reading and manually tagging all reviews collected.

In this sense, Maalej and Nabil [56] tested the performance of Naive Bayes, Decision trees and Maximum Entropy Modeling (MaxEnt for short) algorithms to classify reviews intention as "Bug report", "Feature request", "Rating", or "User experience". As defined by the authors, bug reports describe problems with the app that should be fixed. In feature requests, users ask for additions or changes in terms of features or content. User experiences reflect the experience of users with the app and its features. Finally, ratings are less informative as they simply reflect

the numeric star rating. Therefore, identifying bug reports and feature requests have a direct impact on new requirements.

Similarly, Panichella et al. [65] proposed to classify the intention of reviews as “Feature request”, “Problem discovery”, “Information seeking” or “Information giving” by combining NLP, sentiment analysis and text analysis techniques. While “Problem discovery” is analogue to bug reports, information seeking are reviews containing questions to developers or other users, and information giving inform users or developers about aspects of the app.

Later, Di Sorbo et al. [33] proposed SURF, an approach that classifies reviews according to both intention and topics. Besides adopting the taxonomy introduced by Panichella et al. [65], the authors describe 12 topic clusters, such as “App”, “GUI”, and “Security”. Table 2.1 provides a comparison of the taxonomies found in the literature.

Finally, McIlroy et al [58] selected one-star and two-star user reviews from both Apple App Store and Google Play Store to classify the issues reported according to 14 categories. The taxonomy proposed by the authors offers insights both about user intention (e.g., “feature removal”, “feature request”) and the review topic (e.g., “additional cost to enjoy the app”).

By comparing the different studies, what stands out is that review topics tend to be more domain-specific than intentions. For instance, topics such as “Privacy” or “Security” may only arise from apps where user data is heavily involved. Therefore, predefined taxonomies for topics risk leaving out important insights for requirements engineers.

In this context, feature extraction has been addressed by many authors as a way to produce app-specific, detailed summaries from the reviews. The next section details the main approaches found and what can be learnt from them.

Extracting features

As discussed on Section 2.2.4, evaluating current implementations to support app evolution has become a prominent application of review analytics. By analyzing what users write, requirements engineers are able to judge how well an implement requirement (e.g., a live feature) has reached its original goals, as well as identify alternative ways to fulfill a certain need.

For that, many studies applied NLP techniques to identify relevant app features from the reviews or app descriptions. Harman et al. [44] extracted features from the app descriptions that are publicly available on the app store. Their approach relies on informal patterns used

Paper	Classes	Approaches
McIlroy et al [58]	Additional cost, Functional complaint, Compatibility issue, Crashing, Feature removal, Feature request, Network problem, Other, Privacy and ethical issue, Resource heavy, Response time, Uninteresting content, Update issue, User interface	Naive Bayes Decision trees SVM
Yang and Liang [86]	Functional requirement, Non-functional requirement	Concept dictionary
Maalej and Nabil [56]	Bug report, Feature request, User experience, Rating	Naive Bayes Decision trees MaxEnt
Panichella et al. [65]	Feature request, Problem discovery, Information seeking, Information giving	Naive Bayes SVM Logistic regression J48 ADTree
Di Sorbo et al. [33]	User intentions: Same as Panichella et al. [65]; Review topics: App, GUI, Contents, Pricing, Feature or functionality, Improvement, Updates/Versions, Resources, Security, Download, Model, Company	User intentions: Same as Panichella et al. [65] Review topics: Concept dictionary

Table 2.1: Summary of classification-focused studies

by developers to present the main features, such as lists. Based on these, “featurelets”, i.e., groups of commonly occurring co-located words such as {near, wifi, hotspot}, are created to represent the feature.

Groups of commonly occurring words, also called collocations, are a popular technique in the literature for extracting features from the reviews. Guzman and Maalej [43] used collocations appearing in at least 3 reviews in their fine-grained analysis. The authors also applied tactics to group similar features, such as matching synonyms and applying topic modelling to the results. Keerpati et al. [51] adopted a similar approach, but focused on words of specific POS, such as nouns. Gao et al. [39] identified “phrases” derived from bi-grams to prioritize issues for developers.

Finally, Johann et al. [50] proposed the SAFE framework, which provides an improvement in performance over past methods, with a precision of 70%, recall of 56% and F1-score of 62%. The authors established a list of POS patterns and sentence patterns of different lengths that are frequently used to describe features. Based on such list, features were extracted from both app descriptions and user reviews, and matched against each other using cosine similarity.

Although improvements have been made over the years, fully-automated feature extraction is not yet ready for industrial use. Therefore, as shown in Section 2.2.4, practitioners still rely on manual feature tagging or semi-automated approaches based on custom keyword lists.

Next, we analyze how the existing literature addresses the problem of summarizing reviews, i.e., presenting the most important issues found after processing the reviews.

Summarizing reviews

The literature employs different approaches to summarize the reviews for analysis. The approaches found typically use different levels of aggregation that allow analysts to navigate from high-level topics to features, reviews, and even individual sentences.

On AR-Miner, Chen et al. [27] summarized the reviews by displaying the 10 most important topics (groups of features) found, ranked by an importance score called *GroupScore*. As showed by Figure 2.7, the *GroupScore* considers group volume (where p denotes the group participation in a review, from 0 to 1), group time series (where k is a unity of time from a time period K and l is a monotonic increasing function), and group average rating (based on star rating of reviews).

$$f^G = \{f_{Volume}^G, f_{TimeSeries}^G, f_{AvgRating}^G\}$$

$$f_{Volume}^G(g) = \sum_{i=1}^n p_{r_i g}$$

$$f_{TimeSeries}^G(g) = \sum_{k=1}^K \frac{p(g, k)}{p(g)} \times l(k)$$

$$f_{AvgRating}^G(g) = \frac{f_{Volume}^G(g)}{\sum_{i=1}^n p_{r_i g} \times r_i \cdot R}$$

Figure 2.7: Group ranking equations [27]

Therefore, the group ranking helps surface issues that are urgent (because they are volumous, recent, and/or have a low rating). Figure 2.8 shows a summary produced by this approach. Similarly, Keertipati et al. [51] used three prioritization approaches to produce a summary of the 10 most critical features and their priority score.

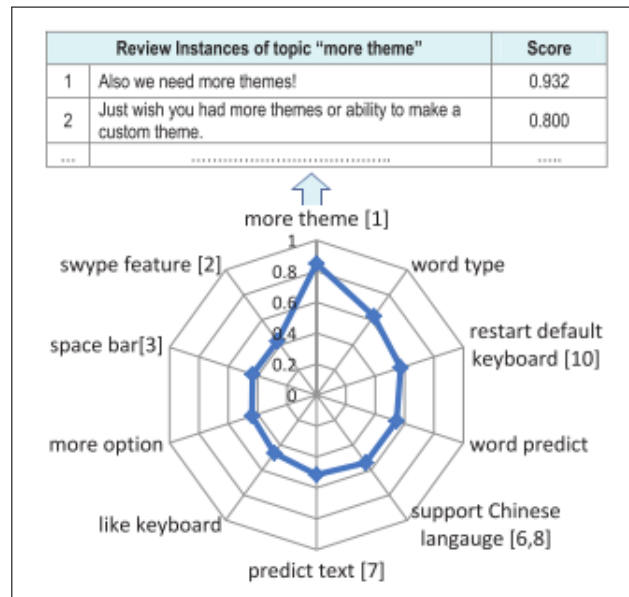


Figure 2.8: AR-Miner summary [27]

Hu and Liu [47] proposed a format to summarize product reviews that could be used in the app store context. The authors presented product features ranked by frequency in the reviews, and then grouped sentences mentioning each feature into positive or negative, as Figure 2.9 illustrates.

Guzman [43] used a two-level summary that shows the frequency and sentiment per topic (groups of features) or per feature. The authors argued that future studies could use this

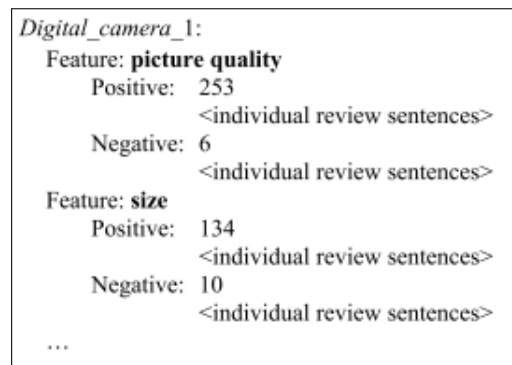


Figure 2.9: Product review summary [47]

format to produce an interactive visualization.

Di Sorbo et al. [33], on the other hand, proposed an interactive summary where users can first browse topics and later intentions to find review sentences describing relevant issues. By inspecting the sentences, the analyst can see the full review it was taken from. Figure 2.10 shows an extract with sentences describing bugs in the user interface. Although the tool was perceived as useful after an experiment conducted by the authors, participants reported that a better navigation, use of visualizations and filters would improve the artifact.

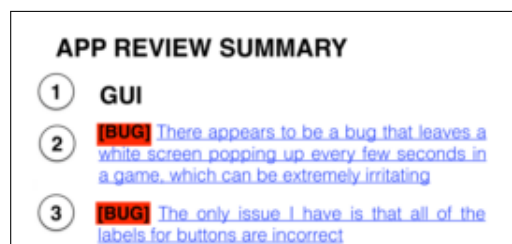


Figure 2.10: Di Sorbo et al. summary [33]

As it turns out, most summaries use topics or features as the initial level of detail, supported by information that helps derive priority, such as priority scores, sentiment, volume, or frequency. Besides that, the studies suggested that showing the full review or excerpts from it as the lowest level of detail allows analysts to better understand the issues being summarized.

Competitive analysis

Another promising application of NLP is to extend the analysis of reviews to competing apps, so to enable insights from the comparison of user opinion across similar apps.

Although this area is still little explored, some contributions have been identified. Jin, Ji and Gu [49] worked on the identification of comparable sentences from the reviews, i.e. extracts where users discuss the same topic or feature. Through feature extraction, sentiment analysis,

similarity functions and clustering, the authors proposed an approach where it is possible to compare user opinion about a topic by analyzing pairs of sentences extracted. For instance, one could read different pairs of sentences discussing aspects of the battery life of two products, and therefore identify areas for improvement.

A previous contribution was made by Fun, Lin et al. [38] where the authors proposed WisCom, a system that enables summarization at review, app, and market level. At the market level, data from competing apps can be analysed in an aggregate manner. The authors showcased the tool by identifying the most common complaints from users about different apps from various categories, such as Education, Games, etc. Although the tool can be used to obtain insights about frequent complaints or praise in a certain market, no direct comparison between apps is performed.

2.4 Information visualization

In this section, the literature on Information Visualization (IV) is studied, with a special focus on its application on Requirements Engineering. We present its definition, guidelines and frameworks for developing IV tools, and what applications already exist of IV in RE.

2.4.1 What is Information Visualization

The use of visualization as a way to improve the communication process is a known issue [66]. From the different visualization paradigms, we are particularly interested on the use of Information Visualization (IV), more specifically on how it can facilitate requirements elicitation. Therefore, we present a definition for IV and distinguish it from other visualization perspectives, namely Data Visualization (DV) and Knowledge Visualization (KV) [13].

- Information visualization (IV): The application of computer supported, visual representations of abstract data to amplify cognition [25].
- Data visualization (DV): graphical representation of unprocessed information (e.g., points, lines or bars) [45].
- Knowledge visualization (KV): an approach that examines the use of visual representations to improve the transfer of knowledge between at least two persons or group of persons [23].

Amplifying cognition means, in practice, that IV relies on the fact that human vision bandwidth is estimated at 100Mb/s, so it can convey more information to the brain than other senses, thus “speeding up” the process of filtering among competing theories about the collected data [37]. Therefore, IV is regarded as a great way to generate insights, and its value is more evident in the following situations [53]:

- When there is a good underlying structure so that items close to one another can be inferred to be similar;
- When users are unfamiliar with a collections contents;
- When users have limited understanding of how a system is organized and prefer a less cognitively loaded method of exploration;
- When users have difficulty verbalizing the underlying information need;
- When information is easier to recognize than describe.

This suggests that IV can be used to help analysts and requirements engineers to understand how users feel about an app or a set of features without having to manually read all reviews up-front. It has the potential to surface topics of interest, and help them navigate the reviews to elicit requirements.

2.4.2 Developing an IV tool

If visualizations are difficult to interpret, a higher cognitive load is placed upon the user, which can diminish the benefits of using IV [66]. Thus, we analyse which visualization principles and frameworks are available to guide the development of new tools.

Visual perception principles

As Fekete et al. [37] discuss, there are two theories that can explain how people perceive features and shapes: Preattentive processing theory [80] and Gestalt theory [52].

According to the preattentive processing theory, a limited set of visual properties are detected very rapidly and accurately by the low-level visual system. A typical example is the detection of a red circle among dozens of blue circles, as showed in Figure 2.11. Just like color, other properties such as line orientation, curvature and line length are also preattentive for some tasks and limits.

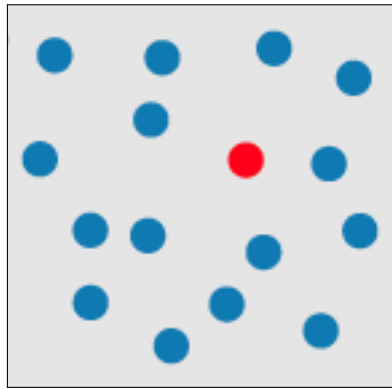


Figure 2.11: Example of preattentive task: detecting the presence of a red circle

When designing an IV tool, such theory can be explored to draw attention to areas of potential interest in a visualization. Besides target detection (as exemplified by the red circle case), other visual preattentive tasks include boundary detection, region tracking, and counting and estimation.

On the other hand, the Gestalt theory offers insights about our high-level cognition when looking at a visualization. It can be summarized by the following principles [82]:

- **Proximity:** Elements tend to be perceived as aggregated into groups if they are near each other;
- **Similarity:** Elements tend to be integrated into groups if they are similar to each other;
- **Continuity:** Oriented units or groups tend to be integrated into perceptual wholes if they are aligned with each other;
- **Symmetry:** Symmetrical components will tend to group together;
- **Closure:** A closed contour tends to be seen as an object;
- **Relative Size:** In general, smaller components of a pattern tend to be perceived as objects.

Interaction principles

Pfitzner et al. [66] proposed an unified framework showing that a visualization design is directly influenced by six factors: data, task, interactivity, skill level, and context, as showed by Figure 2.12.

The data factor concerns the structure and the level of abstraction present in the data being visualized. The task factor relates to what users want to do with the tool and presents several

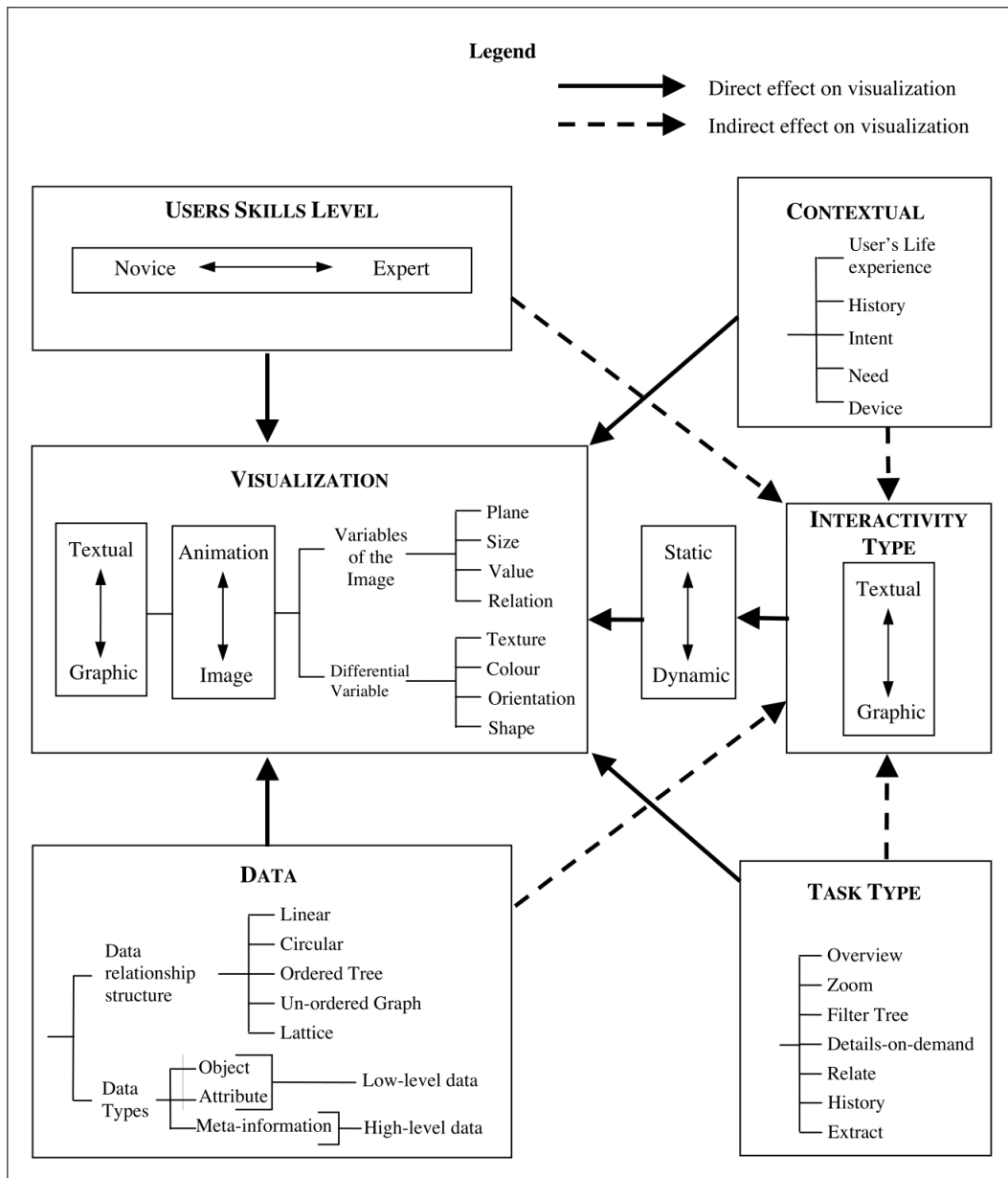


Figure 2.12: IV framework [66]

dimensions by which their goals can be achieved. These dimensions are also recalled as the Shneiderman's mantra [70]: "Overview first, zoom and filter, then details-on-demand".

- **Overview:** a view of the total collection.
- **Zoom:** a view of an individual item. This may be either at the object or attribute level.
- **Filter:** removing unwanted items from the displayed set.
- **Detail-on-demand:** getting the details of a selected group, sub-group or item.
- **Relate:** viewing the relationships between a selected group, sub-group or item.
- **History:** the actions of undoing, replaying, and refining using a store of historic information.
- **Extract:** the extraction or focusing in on sub-collection and other parameters from a given set.

The interactivity factor refers to the manner that information is displayed to the user. The tool will often contain a mix of textual and graphic information, where text allows for a high level of definition but requires significant cognitive load from the user and graphics are easier to analyze and spot patterns [66]. The skill level factor accounts for the fact that as users obtain more experience with the tool, the way they interact with it will also change. Finally, the context factor describes aspects that are external to the use of the artifact, but have an influence on how users will interact with it. For instance, the designer should consider the history aspect, i.e., whether the tool will have an one-off use (e.g., look up something) or will be part of an ongoing activity (such as continuous research on a topic) [66].

One should notice that not only each factor has a direct impact on the visualization, but also one factor may influence others. For instance, in a high-level interaction, where users are interested in obtaining an overview of the data, meta-information can be graphically represented by variables such as size and value. On the other hand, if users want to obtain more detail, lower level data types can be used, and the interaction occurs at the object or attribute level.

2.4.3 IV for requirements engineering

Requirements engineering is considered the most data intensive and media-rich phase of software engineering, where there is a need to define stakeholders, problems and goals [41]. When applied

to RE, visualization helps create a path from data to decision, by playing two central roles. First, it represents the requirements information by highlighting certain constructs and relationships while ignoring others. Second, it serves as the interaction medium to augment a requirements analysts knowledge discovery with advanced computational capabilities [70].

Visual requirements analytics process

Reddivari et al. [70] proposed a framework that characterizes the key components and their interactions in the visual requirements analytics process. Such framework can be used to evaluate existing tools, but also as a guideline to develop new ones. As Figure 2.13 shows, there are five main components: user, data, model, visualization, and knowledge.

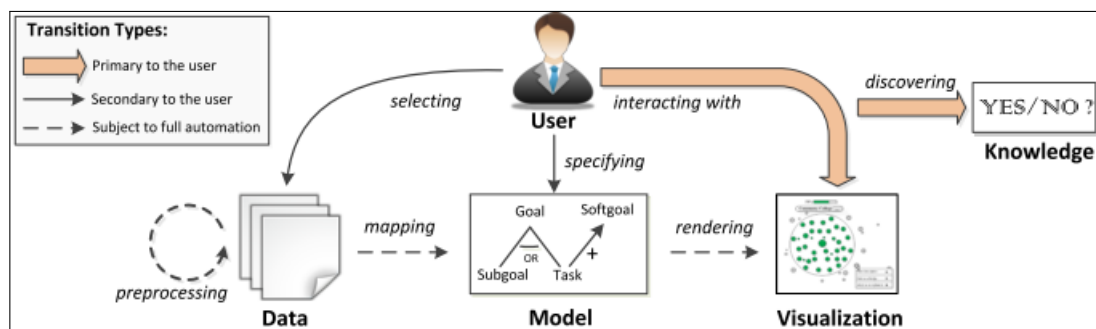


Figure 2.13: Requirements analytics framework [70]

According to the authors, the user is the most important component in the process [70]. It denotes anyone who will be using the tool; e.g., a product manager, an analyst, or a requirements engineer. It is fundamental to understand what are their goals, and which questions they aim to answer by interacting with the visualization.

Next, the selected data needs to be preprocessed in order to extract relevant requirements information for further visual and automatic analyses. Based on the preprocessed data, a model is made defining what entities and relationships will be used to support the users RE task at hand. Among commonly employed models are features, use cases, and goals. Such model is fundamental to map the data into a visualization that serves the user goal.

However, seldom there will be a single requirements visualization that conveys all the necessary information to the user. Instead, the tool developer should create effective and efficient ways to analyze the data, focusing on the interaction rather than only in the visualization [70].

Finally, it is important that by interacting with the tool, the user augments their ability to discover knowledge. This must lead to actionable decision, otherwise the tool will not have achieved its purpose.

State of the art

Visualization has been used in the literature to support various Requirements Engineering tasks, such as detecting ambiguity in requirements [31], understanding risks [35], and navigating interdependencies [72].

Cooper et al. [29] did a survey on the state of the art in terms of visualization for requirements engineering. The authors classified existing studies according to a framework consisting of RE phases and activities. Figure 2.14 shows the framework used, where it is possible to see how the activities are emphasized over the different phases of RE process.

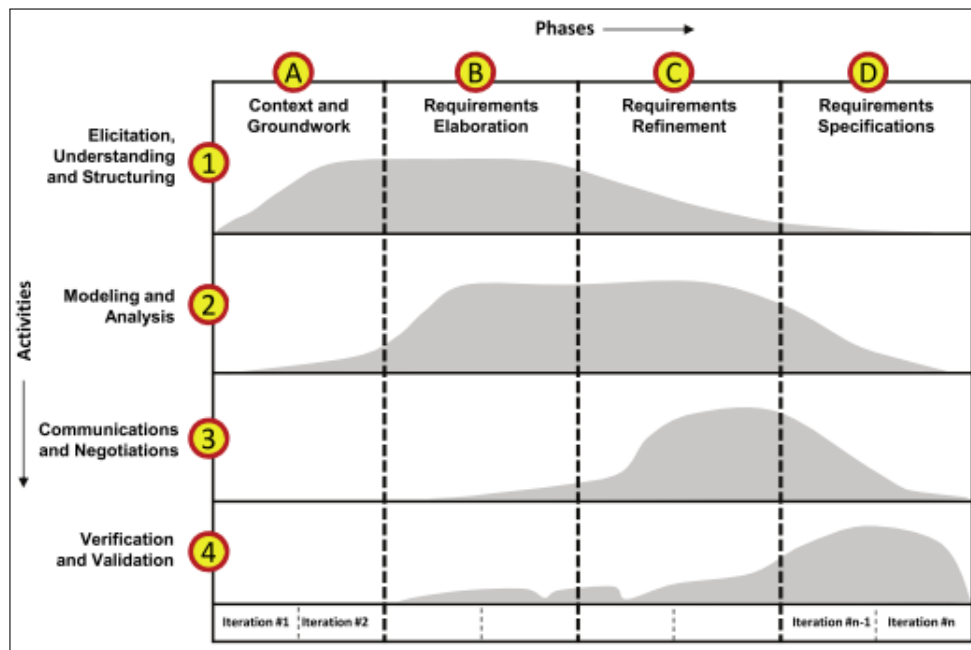


Figure 2.14: RE lifecycle [29]

Besides that, Cooper et al. also categorized the visualizations employed by the studies according to 5 types. Table 2.2 describes and exemplifies each of them.

By positioning the 29 studies found in this framework, the authors showed that most of the efforts of visualization in RE are placed in the activity of modelling and analysis. In terms of RE phases, requirements refinement and requirements specification received most of the attention so far, which revealed the need for more work dedicated to support the early phases of RE.

A total of 5 works employed visualization in the activity of Elicitation, Understanding and Structuring. In the Context and Groundwork phase, Feather et al. [35] used several quantitative/metaphorical visualizations (such as bar charts, radar charts, tree maps) to support early-lifecycle decision in the development of spacecraft technologies. Their approach consists

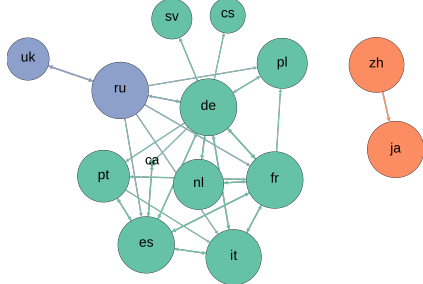
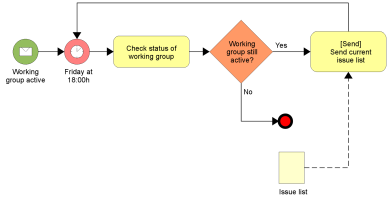
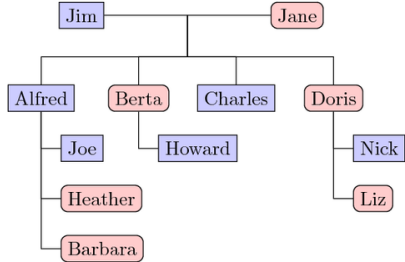
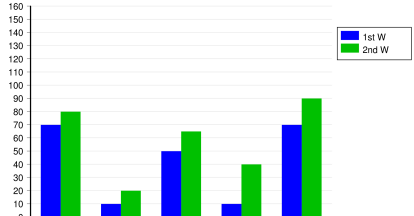
Visualization type	Example																		
<p>Tabular visualizations contain series of intersecting rows and columns that typically hold textual information [29].</p>	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 5px;">7C0</td> <td style="padding: 5px;">hexadecimal</td> </tr> <tr> <td style="padding: 5px;">3700</td> <td style="padding: 5px;">octal</td> </tr> <tr> <td style="padding: 5px;">11111000000</td> <td style="padding: 5px;">binary</td> </tr> <tr> <td style="padding: 5px;">1984</td> <td style="padding: 5px;">decimal</td> </tr> </table>	7C0	hexadecimal	3700	octal	11111000000	binary	1984	decimal										
7C0	hexadecimal																		
3700	octal																		
11111000000	binary																		
1984	decimal																		
<p>Relational visualizations contain a collection of nodes and connectors that describe or indicate a relationship between components or a system, but do not implicitly describe the inherent order of operation of the system [29].</p>																			
<p>Sequential visualizations transmit the order of operation between parts of the system, or of a user and the system [29].</p>																			
<p>Hierarchical visualizations show the decomposition of a system and its parts, as typically used in goal-based modeling approaches [29].</p>																			
<p>Quantitative/Methaphorical visualizations convey relative data and may use visual metaphors and other visual clues such as color, shape, line thickness, and size to convey meaning at a glance [29].</p>	 <table border="1" style="margin-left: auto; margin-right: auto;"> <caption>Data for Quantitative/Methaphorical Visualization</caption> <thead> <tr> <th>Category</th> <th>1st W (Blue)</th> <th>2nd W (Green)</th> </tr> </thead> <tbody> <tr> <td>A</td> <td>70</td> <td>80</td> </tr> <tr> <td>B</td> <td>10</td> <td>20</td> </tr> <tr> <td>C</td> <td>50</td> <td>65</td> </tr> <tr> <td>D</td> <td>15</td> <td>40</td> </tr> <tr> <td>E</td> <td>70</td> <td>90</td> </tr> </tbody> </table>	Category	1st W (Blue)	2nd W (Green)	A	70	80	B	10	20	C	50	65	D	15	40	E	70	90
Category	1st W (Blue)	2nd W (Green)																	
A	70	80																	
B	10	20																	
C	50	65																	
D	15	40																	
E	70	90																	

Table 2.2: Visualization categories [29]

of visualizing requirements, their risks, and possible mitigations to them so that stakeholders could choose among alternate solutions to satisfy a requirement.

In the Requirements Elaboration phase, 4 studies are found by Cooper et al.. Pichler and Rumetshofer [67] analysed the use of a business process modelling visualization to help stakeholders model scenarios from which new requirements were elicited. Beatty and Alexander [18] used tabular visualization to enumerate all possible behaviors of each element in a user interface, in order to support the capture of new UI requirements. The so-called Display-Action-Response (DAR) model proposed by the authors was tested in a technology manufacturing company, and although the model was considered straightforward and easy to use, the main challenge identified was that it relied on a fully manual approach.

Bimrah et al. [21] used relational visualization to model trust relationships as part of requirements elicitation. The modelling language proposed by the authors supports requirements engineers when identifying the relevant actors in the system context and understanding how they trust each other. Finally, Sen and Jain [73] work spans both Requirements Elaboration and Requirements Refinement phases. In the context of goal modelling (a method to represent requirements as goals), the authors used hierarchical and tabular visualizations to elicit and refine soft goals from the involved stakeholders.

Although the survey from Cooper et al. offers a solid overview of research efforts in terms of IV in requirements engineering, it dates back to 2009. In a more recent literature review, Abad et al. [13] showed that requirements elicitation is now the most visualization-supported activity in RE. However, as the authors concluded, there is a clear need for more effort towards interactive visualizations and visualizations that support distributed RE.

State of the practice

Even though IV has not been applied to app review analytics in the literature yet, the two main commercial players in the field, Appfollow and Appbot, do employ some degree of visualization in their tools. Figure 2.15 illustrates a chart in Appfollow [16] where it is possible to see the volume of reviews received by an app over time, per rating given. By hovering over a column, users can check the distribution of ratings for a given day. By looking at this chart, it is possible to obtain an overview of the number and ratings of reviews received in the last days or identify peaks that might be caused by underlying issues that deserve attention by the development team.

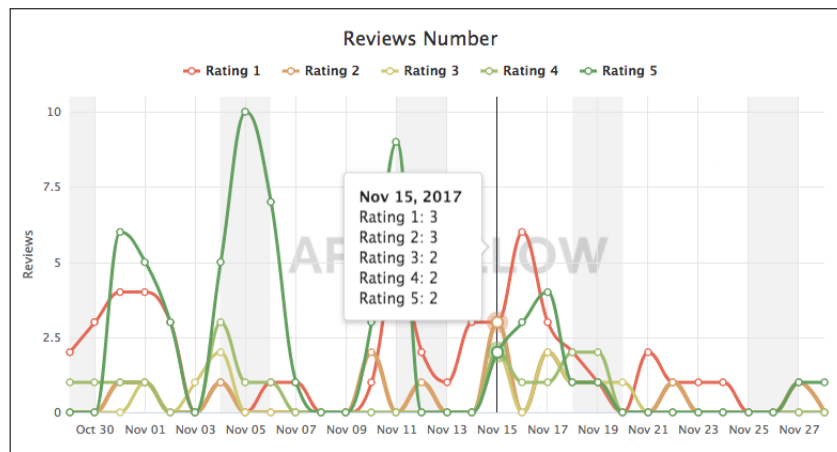


Figure 2.15: Appfollow visualization - Rating over time [16]

An overview of reviews over time is also provided by Appbot, as Figure 2.16 shows. However, instead of ratings, the app version reviewed is plotted in different colors.

The tool also uses other visualizations based on metadata generated by NLP techniques. Figure 2.17 displays a word cloud with the most popular words found in the reviews. The size and position of the words are used to convey frequency, and its color shows the sentiment associated to the word.

Lastly, Figure 2.18 presents a visualization where the reviews can be navigated by their emotion. Axis X represents a continuum that goes from “Displeased” to “Pleased”, and axis Y represents the level of assertiveness detected in the review. Furthermore, the rating given by the user is mapped to the dot color: red for 1-2 stars, orange for 3 stars and green for 4-5 stars [1].

By hovering over the dots, users can read the full review, as well as its date, app version, location, and rating. Besides giving an overview of sentiment and emotion distribution of the reviews, the analyst might want to inspect reviews from a specific area in the chart. For instance, reviews where users are highly assertive and displeased can contain urgent issues, such as crashes or critical missing features.

The framework introduced by Reddivari et al. [70] allows for evaluating such visualizations by answering a set of questions in each of the dimensions *user*, *data*, *model*, *visualization*, and *knowledge*, as shown in Figure 2.19.

As Figures 2.20, 2.21, 2.22 and 2.23 show, the visualizations generally score high in the *user* and *data* factors. One possible explanation is that in such commercial tools, the presence of unintuitive or slow, unefficient interactions might directly affect the success of the business.

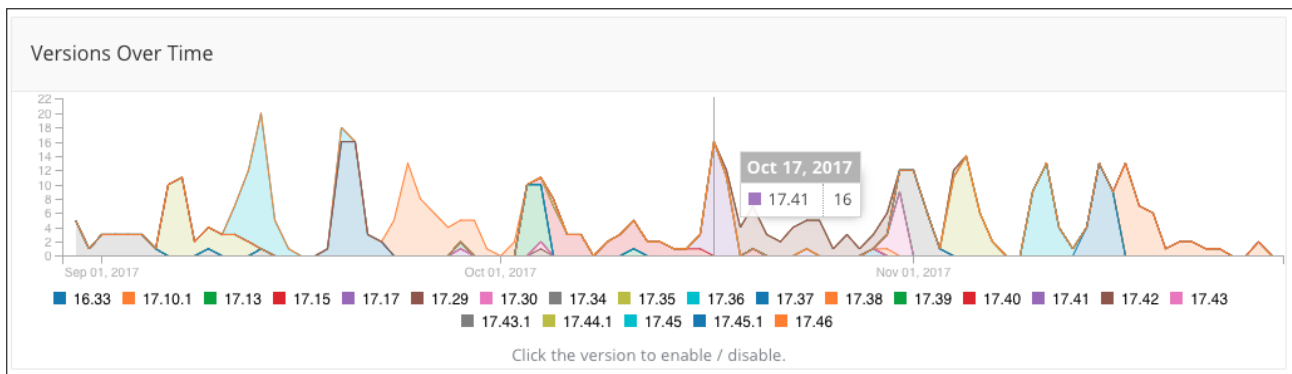


Figure 2.16: Appbot visualization - Versions over time [1]

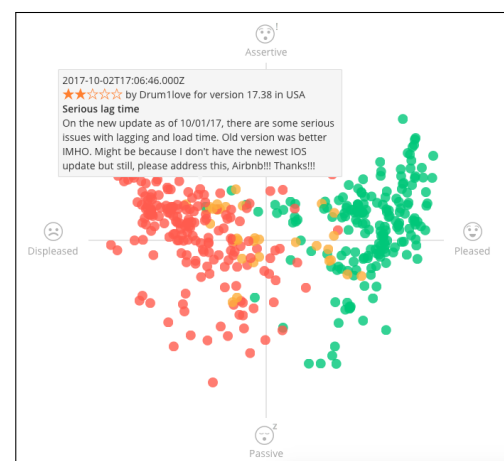
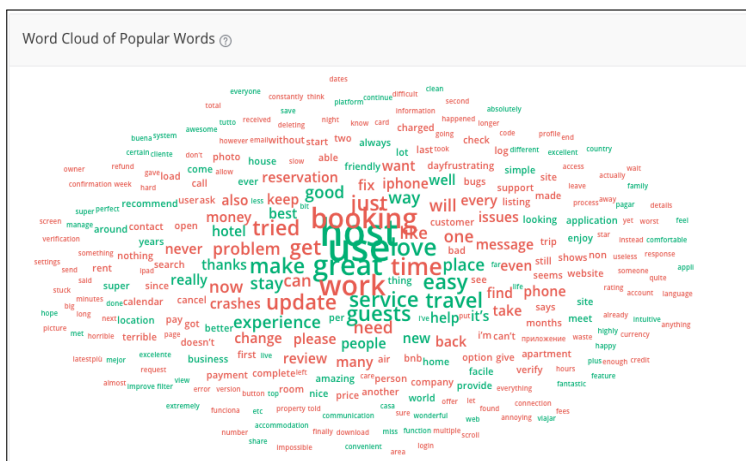


Figure 2.17: Appbot visualization - Word cloud of popular words [1]

Figure 2.18: Appbot visualization - Reviews per emotion [1]

User	Data	Model	Visualization	Knowledge
U1 Multiple stakeholder roles	D1 Large-scale inputs	M1 Explicit model representation	V1 Multiple views	K1 Anomaly detection
U2 Usage without heavy training	D2 Heterogeneous input types	M2 Automatic model construction	V2 Inter-view navigation	K2 Detailed explanation
U3 Real-time performance	D3 Automatic preprocessing	M3 Model extension and customization	V3 Browsing	K3 Hypothesis-based reasoning
U4 Integration into existing software development environment		M4 Model traceability	V4 Searching	K4 Scenario-based reasoning
U5 Practitioner-oriented guidelines			V5 Query-drilling	K5 Actionable decision
			V6 Filtering	
			V7 Annotation	

Figure 2.19: Conceptual goals and their operational questions to be addressed by a visual requirements analytics approach [70]

On the other hand, the *visualization* and *knowledge* factors represent weaknesses for them. Their level of interactivity is very low or inexistent, and all of them consist of static views that cannot be manipulated or zoomed in. Thus, the knowledge discovery is limited, and tends to be restricted to having an overview of the current situation and detecting anomalies potentially

represented by peaks or unusually frequent terms among the reviews.

In the *model* factor, the visualizations offer support in modelling the user satisfaction level according to some parameter, such as version, time, or app aspect/feature. However, most of the times they do not directly represent such models, but rather make it easier for the analyst to do so (when compared to manually reading the reviews).

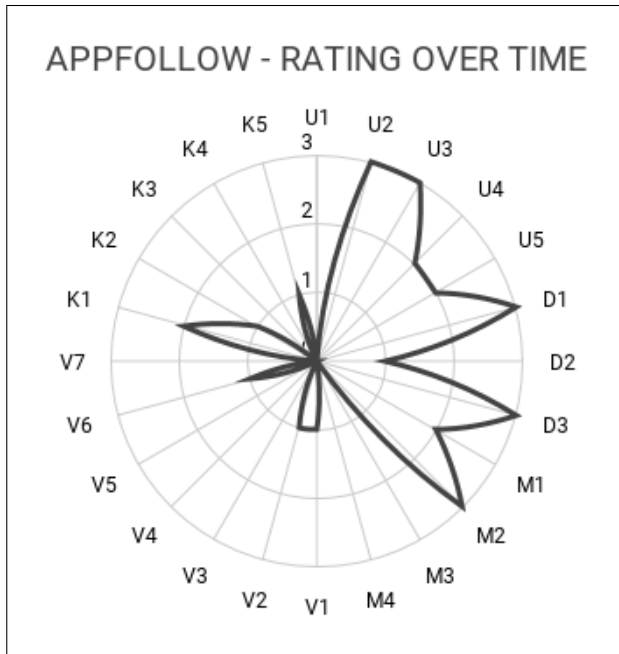


Figure 2.20: Evaluation for Appflow visualization - Rating over time

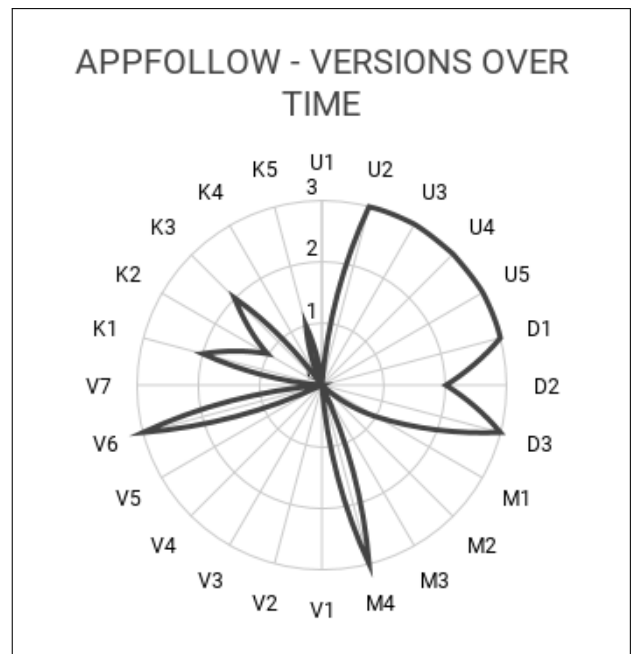


Figure 2.21: Evaluation for Appbot visualization - Versions over time

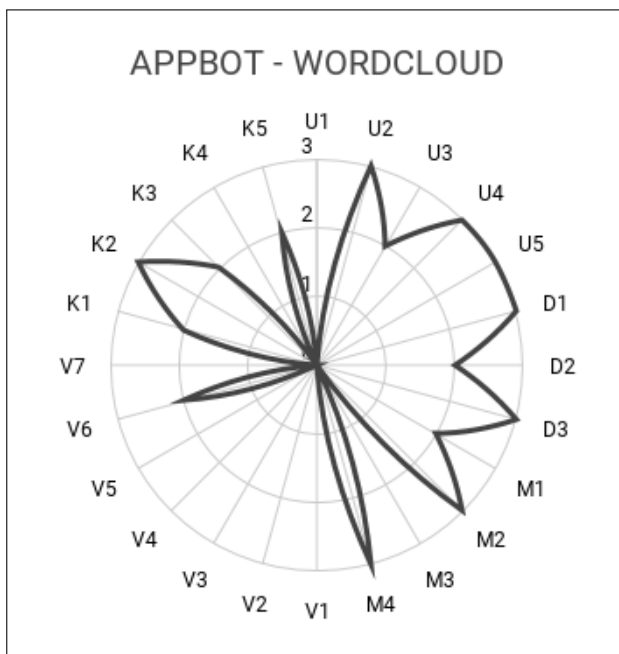


Figure 2.22: Evaluation for Appbot visualization - Wordcloud

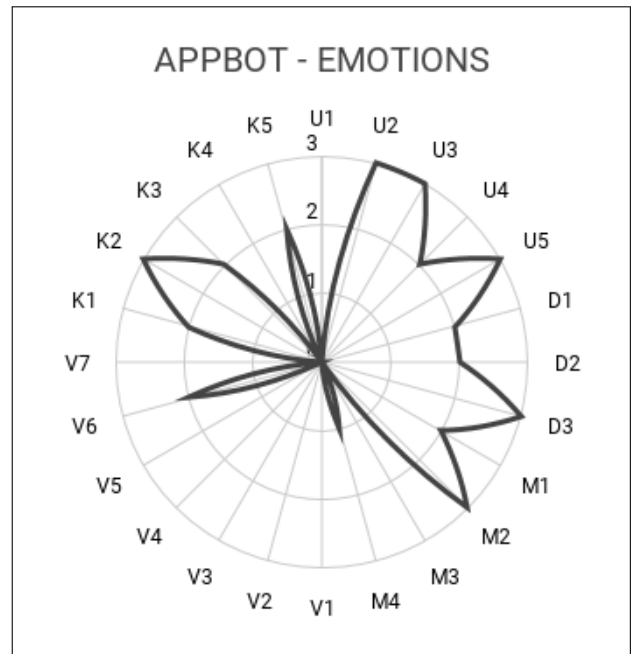


Figure 2.23: Evaluation for Appbot visualization - Reviews per emotion

To sum up, although these applications are rather simple and little interactive, their existence suggest that the benefits of using visualization also apply to the case of crowd-based requirements, in particular to the analysis of app store reviews. However, as no literature on IV addresses this use case, there is room for research evaluating whether visualization support produces a positive impact for eliciting requirements from user reviews.

2.5 Literature findings

In this section, we summarize the findings from the literature review and highlight which gap the proposed thesis aims to fill. We also present the milestones for the key phases of the project.

The literature review on Crowd-based Requirements Engineering revealed that app store reviews are an abundant source for requirements elicitation, from which analysts can identify bugs, feature requests, and potential improvements from a large, distributed audience. As a paradigm shift occurs in the Requirements Engineering process, product development teams need new tools and techniques to support them in making real-time decisions based on user feedback data. Such tools must be able to systematically filter out noise and offer actionable insights about which parts of the app should be improved and why.

Current Natural Language Processing techniques can contribute to this problem by offering automated means to do preprocessing, text classification, feature extraction, and topic modelling. In fact, such techniques have been extensively applied in the literature to support the elicitation of requirements for mobile apps, allowing analysts to have a high-level understanding of the content of reviews received by their app without having to manually read them. However, although app developers can easily access their competitor's reviews, little attention was given in the literature to how such data can be analyzed to influence future product development.

On the other hand, the literature shows that Information Visualization can be applied to RE as a way to augment an analyst's cognition and shorten the path from data to decision. When certain principles and guidelines are followed, an IV tool can speed up the knowledge discovery process, thus benefiting the analyst work. While the application of Information Visualization to requirements elicitation has increased over the last decade, previous studies typically address traditional Requirements Engineering scenarios, and IV support to crowd-based elicitation remains unknown.

In this context, we propose to research how NLP and IV can be used to elicit requirements

from the analysis of user reviews about directly competing apps. We argue that new product opportunities can be potentially uncovered by analyzing competitors weaknesses and strengths from users perspective, and yet no literature exists about the topic. For that, a novel approach, hereby called “RE-SWOT” needs to be defined.

Chapter 3

Tool development

Following the Design Science methodology discussed in Section 1.3, this chapter presents the development of a framework and a tool aimed at facilitating requirements elicitation from app store reviews of competing apps. At one hand, Section 2.2.2 showed that the market landscape and the existing competitors are important sources for gathering requirements for driving software evolution. On the other hand, past contributions on the literature focused on the identification of requirements from reviews of single apps. Although these approaches can be applied by development teams to analyze individual competitors as well, they are not intended to support competitor analysis. They analyze the apps in a siloed manner, and give little attention to how a market perspective could lead to meaningful requirements.

To fill this gap, we propose the “Requirements Engineering SWOT” (RE-SWOT), which offers a way to elicit requirements by identifying strengths, weaknesses, opportunities and threats of an app, as revealed by user reviews about such app and its direct competitors. The conceptual framework of RE-SWOT and the general architecture of the tool created to support it are described in Section 3.1. Then, Sections 3.2 and 3.3 show in detail how NLP and IV are used, respectively, to support the activity.

3.1 Competitive analysis for requirements elicitation

3.1.1 Conceptual approach

Our conceptual approach is adapted from the SWOT analysis, a traditional framework for strategic planning. SWOT analysis gives an overview of how a product or business is positioned,

vis à vis its external environment [24]. During SWOT analysis, four factors are identified: strengths, weaknesses, opportunities, and threats.

As defined by Slotovitch et al. [77] and illustrated on Figure 3.1, strengths are enhancers to desired performance while weaknesses are inhibitors to desired performance, with both being internal to the organization. Opportunities are enhancers and threats are inhibitors to desired performance, but these are considered external factors which are outside the organization's control.

	Internal	External
Enhancers	Strengths	Opportunities
Inhibitors	Weaknesses	Threats

Figure 3.1: SWOT matrix

During the creation of SWOT, one should always consider the external environment of the organization. Positive internal aspects might only be considered strengths if they provide an advantage over competitors, e.g., if all players can equally offer a good price, then good prices cannot be considered a strength to any of them. Therefore, comparing the organization to the competition is of vital importance to identifying strengths, weaknesses, opportunities and threats.

Upon identifying these four factors, stakeholders gain a strategic understanding about their business, and have the means to develop an action plan to improve their competitiveness – by improving their weaknesses, exploiting opportunities, sustaining strengths, and preventing threats.

When applying SWOT analysis to a Crowd-based Requirements Engineering scenario, we are mainly interested in analyzing user feedback and its metadata to create a SWOT matrix that can lead to new requirements. Figure 3.2 illustrates the RE-SWOT matrix, which adapts the original SWOT matrix to the problem at hand. A feature represents a strength to an app if such app is performing positively above average when compared to other apps that offer

the same feature. Similarly, a weakness is a feature or issue where the app is performing negatively below the average when compared to the competition. When considering external factors, threats are features from competitors that are performing positively above the market average, and opportunities emerge when a competitor has a feature or aspect that is performing negatively below the market average.

		App with the feature	
		Own app	Competition
Feature performance	Positive and Unique or above market average	Strengths Features that should be kept and/or further expanded.	Threats Well-received features from the competition that could be imitated.
	Negative and Unique or below market average	Weaknesses Issues or bugs that should be fixed or minimized.	Opportunities Gaps at the competition that could be exploited by launching new features.

Figure 3.2: SWOT analysis adapted to Crowd-based Requirements Engineering

For an objective comparison among apps, we quantify feature performance as a combination of (i) user sentiment towards the feature and (ii) number of user reviews mentioning the feature. The resulting score, hereafter called Feature Performance Score (FPS), is introduced and illustrated on Section 3.1.1. By understanding the market average score for a feature, we can determine which players are performing above or below average and therefore assign strengths, weaknesses, threats and opportunities to a certain app. Once the SWOT is generated, the requirements engineer can derive new requirements aimed at keeping or improving the competitiveness of such app. Figure 3.3 summarizes this approach step by step.

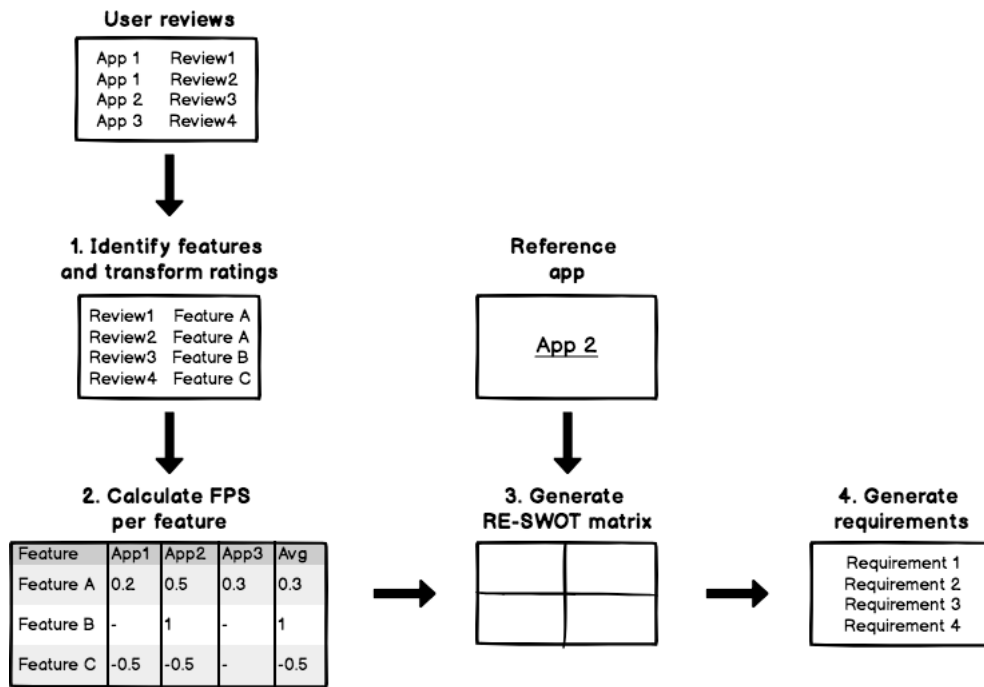


Figure 3.3: RE-SWOT method

Step 1. Identify features and transform ratings

In this step, app features are identified from the user reviews. Our approach to this task is facilitated by NLP techniques, as further described on Section 3.2.2. Furthermore, the original user ratings (on a scale from 1 to 5) are transformed to a sentiment scale ranging from -2 to +2. The objective of this transformation is to assign a positive or negative orientation to the reviews before the FPS scores are calculated.

Step 2. Calculate FPS per feature

Given a set of apps $A = \{a_1, a_2, a_i, \dots, a_m\}$ and a set of features $F = \{f_1, f_2, f_j, \dots, f_n\}$, the feature performance score of app i in relation to feature j can be calculated as presented on Equation 3.1.

$$FPS_{i,j} = \frac{S_{i,j} \cdot V_{i,j}}{\sum_{i=1}^m |S_{i,j} V_{i,j}|} \quad (3.1)$$

Where:

- $S_{i,j}$ (user sentiment for feature j from app i): sum of the transformed user ratings given to the reviews mentioning the feature, divided by the maximum possible sum. For

instance, if a feature was mentioned on two 5-star reviews and one 2-star review, the feature sentiment score for such feature is equal to $(2 + 2 - 1)/(2 + 2 + 2) = +0.5$.

- $V_{i,j}$ (**feature volume for feature j from app i**): number of user reviews from app i that mention feature j . For example, if an app has received 2 reviews “App crashes when uploading photos. Every time I try to upload my photos, there is an error message.” and “App is crashing a lot recently”, the feature volume for *upload photos* is 1.

Step 3. Generate RE-SWOT matrix

Based on the FPS scores from step 2, a RE-SWOT matrix can be generated. For each feature, the scores for each app are evaluated according to three criteria: first, whether they are positive or negative, second, if they are above or below the average score for that feature, and third if the app is a competitor or not.

The first criteria is described by Equations 3.2 and 3.3. A FPS is considered positive if Equation 3.2 holds true, or negative if Equation 3.3 is true. Consequently, a FPS is said to be neutral when it is within the range $[-\sigma, +\sigma]$.

$$FPS_{i,j} \geq \sigma \quad (3.2)$$

$$FPS_{i,j} \leq -\sigma \quad (3.3)$$

The FPS is also used to determine if a feature is unique, above or below market average. A FPS is above market average if Equation 3.4 holds true, or below average if Equation 3.5 applies. Finally, a FPS can assume values of 1 or -1 if the feature is unique, i.e., no other app in the market as it.

$$FPS_{i,j} - \overline{FPS}_i \geq \sigma \quad (3.4)$$

$$FPS_{i,j} - \overline{FPS}_i \leq -\sigma \quad (3.5)$$

We adopt $\sigma = 0.1$ for Equations 3.2, 3.3, 3.4 and 3.5.

The third criteria evaluates whether the score refers to a feature from the competition or not. Features from the competition that perform positively above the market averages are classified as threats. On the other hand, if they are negative and below the average, they represent opportunities. If the score refers to a feature from your own app, it can be a strength (when

positively above the average) or a weakness (when negatively below the average). Any other features that do not fit the aforementioned scenarios are not included in the RE-SWOT.

Step 4. Generate requirements

After step 3, a list of strengths, weaknesses, threats and opportunities is identified. The last step consists in generating requirements to tackle these. In this step, we apply the TOWS framework [83], which incorporates the result of a SWOT analysis to identify strategies to improve a company's competitiveness.

The TOWS Matrix indicates four conceptually distinct alternative strategies, tactics or actions that can be pursued. In practice, some of the actions identified might overlap or be executed concurrently [83].

We restrict the TOWS matrix to the context of Requirements Engineering, therefore providing a framework to the identification of four types of requirements.

- **WT requirements:** requirements aimed at minimizing weaknesses to make them less susceptible to threats.
- **WO requirements:** requirements aimed at overcoming weaknesses to pursue opportunities.
- **ST requirements:** requirements aimed at using strengths to reduce vulnerability to threats.
- **SO requirements:** requirements aimed at pursuing opportunities that are a good fit to the strengths.

As discussed by Weihrich [83], the TOWS Matrix refers to a particular point in time. However, external and internal environments are dynamic; some factors change over time while others change very little. Therefore, the requirements engineer must consider the time dimension and repeat their TOWS analysis over time to keep their product competitive.

3.1.2 Example

As an example of the technique, we consider a fictional set of user reviews about three photo edition apps, listed on Table 3.1. We assume the perspective of app Photo1, thus considering

Photo2 and Photo3 as competitors. As a result of step 1, the features mentioned on the reviews are identified and the user ratings are transformed.

App	Review	Rating	Transformed rating	Features
Photo1	I love the free filters, I use this app to edit photos everyday.	5	+2	filters edit photos
Photo1	Filters are just great, they give my photos a nice look in a few seconds. Best photo editor!!	5	+2	filters
Photo1	Saving photos is not intuitive at all, it's annoying how you need to click twice to do it. Please improve it!	1	-2	save photos
Photo2	Syncing with the desktop version makes it so easy to edit photos while travelling!	4	+1	syncing edit photos
Photo2	This is the only app I use for editing photos. I'm a professional photographer who is used to the desktop version, so syncing my photos is just great.	5	+2	edit photos syncing
Photo2	App crashes whenever saving a photo, and now you want me to pay for the filters?? Ridiculous.	1	-2	save photos filters
Photo3	I used to love this app because of the advanced exporting capabilities, now it just crashes!!! Terrible.	2	-1	exporting
Photo3	What happened with the last update?? Can't export my photos anymore, simply useless.	1	-2	exporting
Photo3	I always use the filters to edit and save my photos for social media.	4	+1	edit photos filters save photos
Photo3	How can I switch back to the old version?	3	0	-

Table 3.1: Example review set

Figure 3.4 illustrates the outcome of step 2. The FPS scores are calculated for all the features, and all information required for the RE-SWOT generation is ready.

		Photo1	Photo2	Photo3	Avg. FPS
		Reference	Competitor	Competitor	
filters	feature sentiment	+1.00	-1.00	+0.50	+0.14
	feature frequency	2	1	1	
	FPS	+0.57	-0.29	+0.14	
	Situation	Positive, above average	Negative, below average	Positive, on average	
edit photos	feature sentiment	+1.00	+0.75	+0.50	+0.33
	feature frequency	1	2	1	
	FPS	+0.33	+0.50	+0.17	
	Situation	Positive, on average	Positive, above average	Positive, below average	
save photos	feature sentiment	-1.00	-1.00	+0.50	-0.40
	feature frequency	1	1	1	
	FPS	-0.40	-0.40	+0.20	
	Situation	Negative, below average	Negative, below average	Positive, above average	
syncing	feature sentiment	-	+0.75	-	+1.00
	feature frequency	-	2	-	
	FPS	-	+1.00	-	
	Situation	-	Positive, unique	-	
exporting	feature sentiment	-	-	-0.75	-1.00
	feature frequency	-	-	2	
	FPS	-	-	-1.00	
	Situation	-	-	Negative, unique	

Figure 3.4: Illustrative example of step 2

As it can be seen on Figure 3.4, some of the features are not going to be included on the SWOT matrix. Only features that are unique, positively above average or negatively below average are considered.

Based on such calculations, the RE-SWOT of Figure 3.5 is generated. Photo1 has one strength and one weakness, but multiple threats and opportunities. The next step is to use the RE-SWOT to generate requirements.

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

Figure 3.5: Illustrative example of step 3

We start step 4 by identifying **SO requirements**. For this type of requirement, the goal is to

find a fit between opportunities and strengths. Since the filters are a strength for Photo1, there is one opportunity that represents a clear fit: the fact that Photo2's filters are negative and below market average. By reading Photo2's reviews mentioning this feature, we understand that users are dissatisfied about having to pay for filters. Therefore, a possible requirement to pursue this opportunity would be to offer more free filters that could attract these dissatisfied users.

Next, we identify **ST requirements**. From the existing threats, Photo2's edit photos is the only one that could be potentially minimized by Photo1's strength. However, by inspecting Photo2's reviews mentioning this feature, it becomes clear that the reason why users like to edit photo with Photo2 is the syncing capability. Since there is little fit between syncing and filters, we do not generate any ST requirements.

To generate **WO requirements**, we compare Photo1's weakness (save photos) with its opportunities. By improving the way users save photos, Photo1 could potentially pursue two opportunities: first, the dissatisfaction of Photo2's users about saving photos, and second, the dissatisfaction of Photo3's users about exporting photos. Not only Photo1 could improve the usability of saving photos, but also they could consider offering more advanced capabilities, such as saving photos in different formats. This could attract Photo3's users who liked its advanced exporting but are unhappy about recent crashes.

Finally, to identify **WT requirements**, we match weaknesses and threats. The fact that Photo3's users use the app to save photos for social media could threaten Photo1. So a possible WT requirement would be for Photo1 to imitate this capability by offering a similar integration in their app.

Table 3.2 summarizes all the requirements identified by this analysis. Each requirement has a goal and a potential impact, identified from the number of user reviews that justify such requirement. The latest could be combined with other information, such as estimated effort from the team, to prioritize the requirements.

3.1.3 Tool architecture

This section describes the architecture of the RE-SWOT tool, which was implemented with R and Tableau Software and is available online ¹.

¹<https://mparente.shinyapps.io/ReSWOT/>

Requirement	Goal	Potential impact
Improve usability of saving photos	Eliminate weakness to pursue the attraction of Photo2 and Photo3 audience	4 users
Offer advanced exporting options	Attract Photo3's audience	2 users
Offer export to social media option	Attract Photo3's audience	1 user
Offer more free filters	Attract Photo2's audience	1 user

Table 3.2: Illustrative example of step 4

Context viewpoint

The conceptual approach previously discussed is supported by a tool whose context is illustrated by Figure 3.6. The tool is designed for the analysis of app store reviews, which are posted by users online and obtained by the requirements engineer. The uploaded review data is thereafter processed through NLP techniques and displayed in an interactive visualization. Upon manipulating this visualization, the requirements engineer is able to create a RE-SWOT of the app, and derive requirements to improve the current situation.

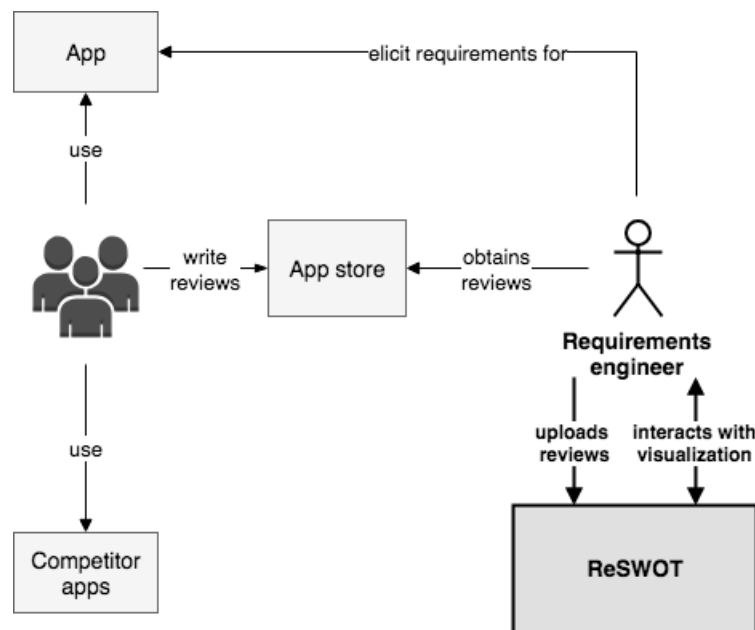


Figure 3.6: Tool architecture - Context viewpoint

Functional viewpoint

A high-level functional viewpoint is shown on Figure 3.7. There are mainly two modules: a NLP module, which is implemented in R [69] through the libraries Udpipes [78], Tidytext [74]

and Quanteda [19], and a visualization module which relies on Tableau software toolset [10]. Both modules are exposed to the requirements engineer through a Shiny [71] web application.

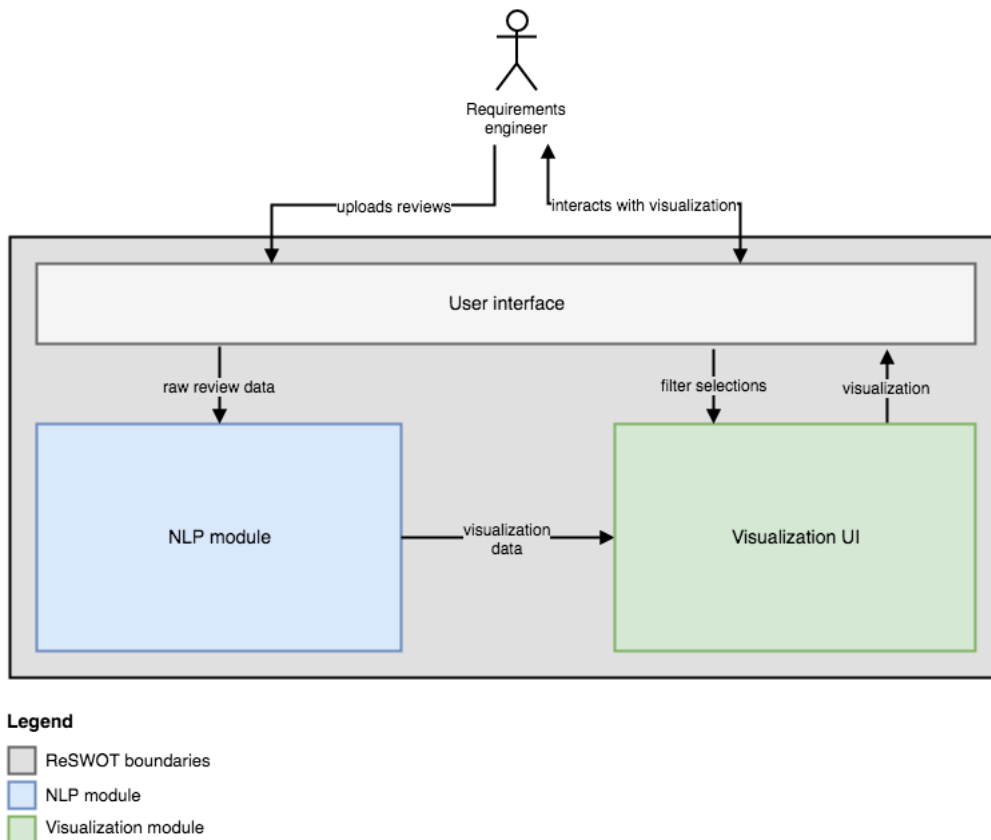


Figure 3.7: Tool architecture - High-level functional viewpoint

The NLP module is responsible for several tasks such as stopword removal, POS tagging, feature extraction, and sentiment attribution. On the visualization module, simple calculations (such as frequencies) are made upon user input, and the final visualization is rendered.

On the NLP module showed on Figure 3.8, the raw review data uploaded by the user is first preprocessed for noise reduction. The original ratings are also transformed into a scale with positive or negative values, as described on Section 3.1.1.

Besides that, the feature extraction component identifies app features mentioned by users on the reviews. The list of identified features is then grouped so that synonyms and similar features are unified. Finally, the reviews are tagged, i.e., the original sentences where the features were mentioned are identified, and all metadata is combined together and stored for later use by the visualization module.

On the IV module illustrated on Figure 3.9, the final visualization is dynamically rendered based on user input. The data previously processed by the NLP module is provided through a live connection and depending on the period chosen and the main app under analysis, the

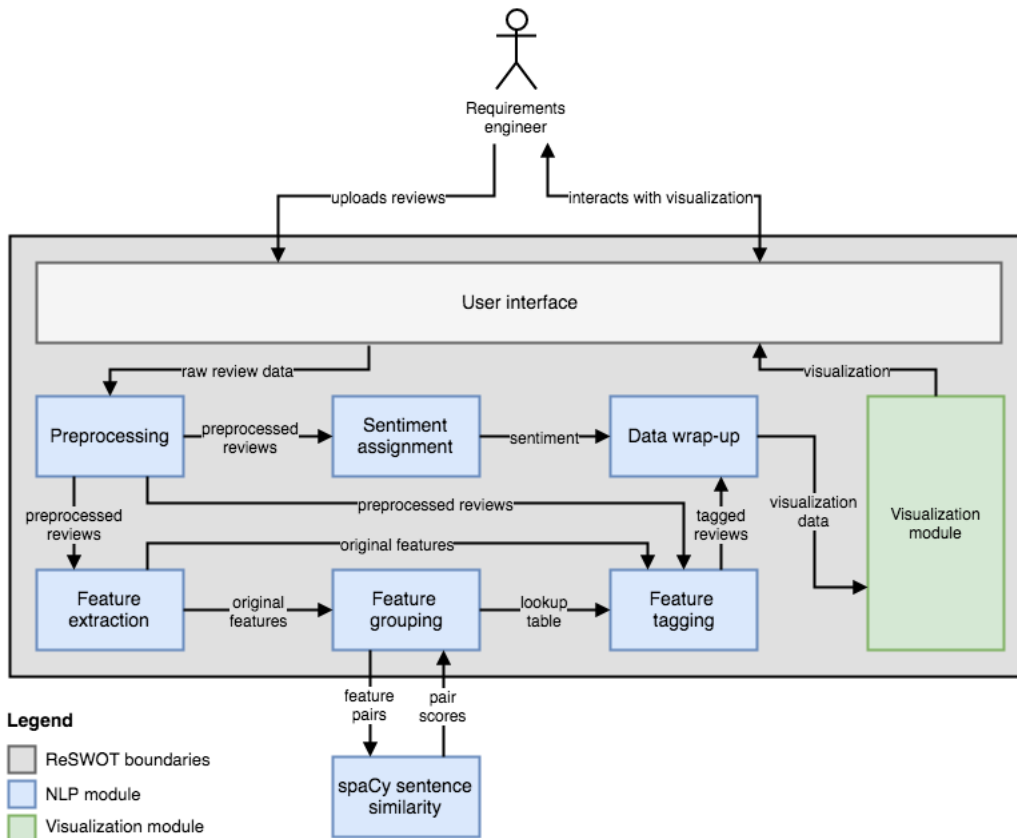


Figure 3.8: Tool architecture - Functional viewpoint of the NLP module

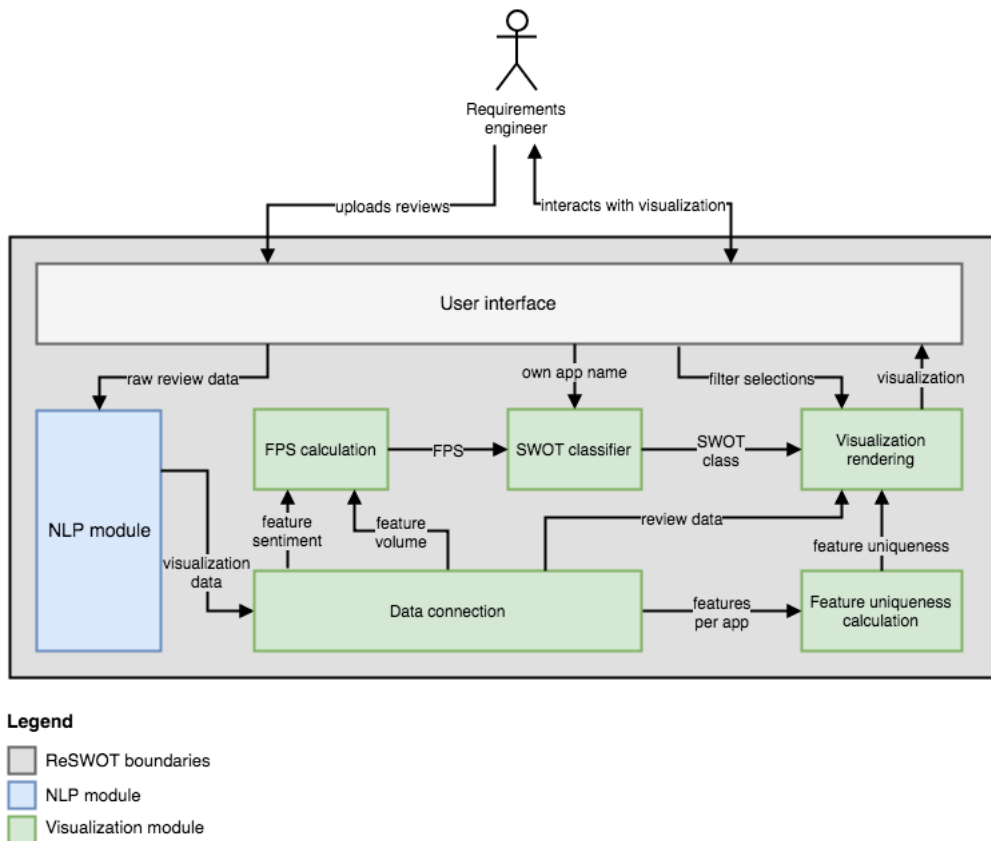


Figure 3.9: Tool architecture - Functional viewpoint of the visualization module

FPS scores are calculated and features are classified as strengths, weaknesses, opportunities, or threats. Based on how unique each feature is (i.e., how many apps contain that feature), the requirements engineer can also directly compare the scores of common features across apps. The interaction between the requirements engineer and the tool, as well as the visualization approach used, are described on Section 3.3.

3.2 Competitive analysis for requirements elicitation via NLP

Section 3.1 introduced a conceptual approach to identify new requirements from the competitive analysis of app store reviews, and the architecture of the corresponding tool. Central to this approach is the identification of app features from the app store reviews, which is implemented through NLP techniques. In this section, we detail which steps are taken to identify and standardize such features before the data is ready for use by the visualization module.

3.2.1 Preprocessing

As a first step we preprocess the inputted user reviews. The feature extraction requires the following preprocessing steps:

- **Tokenization:** We use R library `Udpipe` [78] to tokenize the text into sentences and words.
- **Lowercase transformation:** All tokens are transformed into lowercase for uniformization.
- **Stopword removal:** We use the standard stopwords list provided by `tm` [36] package to remove common English words that are not meaningful for feature extraction. Additional words such as the app name, “feature”, and “app” which are commonly found on the reviews are also removed.
- **Noun, verb, and adjective extraction:** As suggested by Guzman and Maalej [43], features are more likely to be described through nouns, verbs, and adjectives. We use `Udpipe`’s POS tagging [78] capability to identify and filter tokens that follow these criteria.
- **Lemmatization:** We apply `Udpipe`’s lemmatizer [78] to the tokens so that words such as “photos” and “photo” are reduced to the common term “photo”. This step reduces the

noise produced by words with same meaning, which would otherwise result in redundant features.

3.2.2 Feature extraction

As discussed on Section 2.3.3, previous studies found that features are often mentioned through groups of co-occurring words, also called collocations. We use Udpipes's collocation finding algorithm [78] to identify pairs of words that occur unusually often in the reviews of each app. Then, the collocations are filtered so that they only contain words that appear at least 3 times on the reviews. We additionally exclude collocations that follow the patterns (adjective, adjective), (adjective, verb), and (verb, verb), since these did not generate meaningful features during the feature extraction step, as revealed by a manual inspection of the reviews.

3.2.3 Feature grouping

Because users adopt different words to refer to the same features, a grouping step is needed to merge similar features, for example “photo edition” and “edit picture”. Merging similar features is also important to enable the comparison of features in common among different apps.

To identify pairs of similar features, we use Cortical.IO API [30] for calculating the cosine similarity between all combinations of features, regardless of the app where they were identified. Pairs of features with a cosine similarity score equal to or higher than 0.60 are merged together. When doing so, we adopt the name of the feature with the highest frequency, i.e., if “photo edition” appears in 50 reviews and “edit picture” is mentioned in another 30 reviews, then the name “photo edition” is used. A lookup table with the original features identified and their final names is then used for feature tagging.

3.2.4 Feature tagging

The goal of feature tagging is two-fold: first, we want to add metadata to the app reviews so that it is possible to make aggregated calculations such as the feature sentiment, feature volume, and the feature impact score. Second, we want to identify in which sentences the features are mentioned, so that requirements engineers can quickly inspect user feedback when interacting with the visualization.

Based on the lookup table generated by the previous step, we use regular expressions to match

the original feature name to the preprocessed reviews. All the sentences where the feature is mentioned are identified, and the reviews are tagged with the feature name originated from feature grouping. As an example, if one of the reviews contains the sentence “*I use this app to edit all of my pictures*”, it is tagged as mentioning the feature “photo edition”, since “edit picture” and “photo edition” have been previously grouped under the same name.

3.3 Competitive analysis for requirements elicitation via IV

This section describes how Information Visualization is used to support the approach described on Section 3.1, according to the framework of Pfitzner et al. [66]. In particular, the data and task factors comprehensively cover the main characteristics of the tool.

3.3.1 Data factor

The first perspective of the IV framework is the data factor. Table 3.3 describes the data types used for the visualization, namely the objects and their attributes. There are three objects: reviews, features, and apps.

A review has several attributes such as date, title and rating. These allow for contextualizing user feedback, i.e., understanding when an issue was experienced and how unsatisfied the user is. Second, features can be identified from the app reviews. Examples are *use filter* and *save photo* for a photo editing app. The attributes of a feature are its name, app, FPS score and feature volume (which corresponds to the number of reviews mentioning the feature in an app). Finally, the third object is the app, which can be classified as “own app” or a competitor app, depending on who is using the tool. For instance, if Photo1 is taken as reference, then Photo2 and Photo3 are competitors. At the beginning of their interaction with the tool, users can specify which of the apps is their own app, and thereafter the others are classified as competitors.

Besides data types, it is important to understand which relationships exist among the objects. Table 3.4 describes the two relationships found in the data used for visualization. First, mention(feature, review) refers to the fact that a feature can be mentioned in one or more reviews. This fact allows for the calculation of the FPS score, and other meta-information such as feature frequency per rating, and feature sentiment. It is possible that some reviews do not mention any feature. However, these reviews cannot be interacted with on the visualization.

Object	Attributes	Example
Review	Id Content Rating Date App reviewed	review0456 I love this app, I always <i>use the filters</i> before <i>saving my photo</i> for social media. 5 25/06/2017 Photo2
Feature	Name App FPS score Feature volume	save photo Photo2 0.5 15
App	Name Own app/competitor	Photo2 Competitor

Table 3.3: Data types

Relationship	Example	Description
mention(feature, review)	<i>filters</i> is mentioned on review0456	A feature can be mentioned on one or more reviews.
SWOT(feature, app)	Photo2's <i>filters</i> are a threat to Photo1	An app's feature can be a strength, a weakness, an opportunity or a threat to all apps.

Table 3.4: Data relationships

Second, SWOT(feature, app) describes whether a feature is a strength, weakness, opportunity, or threat to an app. Such relationship depends directly on whether the app is a competitor or not. For instance, if Photo1 is taken as a reference, Photo2's filters are a threat. However, if Photo2 is the reference app, then filters can be considered a strength to Photo2. Such relationship is always drawn between a feature and the reference app.

3.3.2 Task factor

As introduced on Section 2.4.2, the task factor can be described according to Shneiderman's mantra and consists of the dimensions overview, zoom, filter, detail-on-demand, relate, history, and extract. The interactions used for each of these dimensions are described in the following sub-sections.

Overview

In this dimension, the goal is to show the user an overview of the total collection they are analyzing [66]. Figure 3.10 illustrates the overview of the features identified from all apps. .

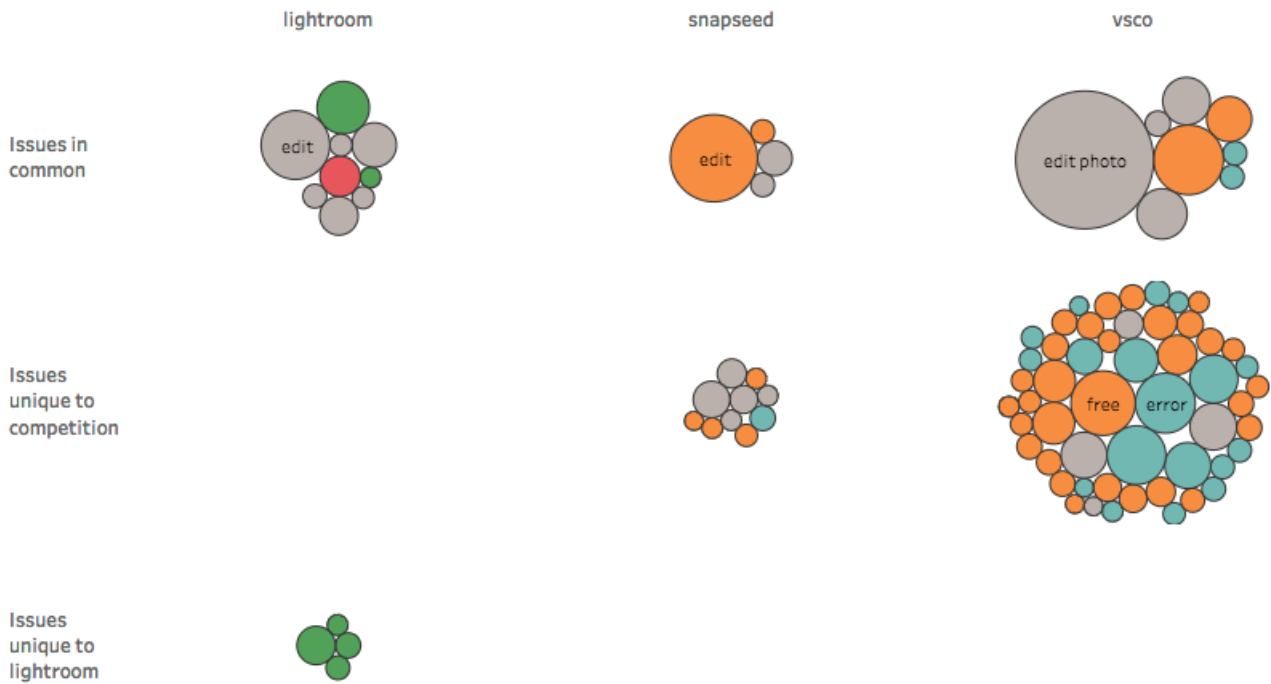


Figure 3.10: Overview of features

Each circle represents a feature. Its position on the x-axis represents the app where the feature was identified, and its position on the y-axis represents how unique the feature is. The size of the circle indicates how frequently the feature is mentioned on the reviews, and its color illustrates whether it is a strength, weakness, opportunity, or threat to the main app under analysis.

Zoom

The visualization allows to zoom into a feature from the collection by clicking on it. When doing so, the visualization is updated to only show the selected feature across any apps where it is mentioned. Figure 3.11 illustrates this concept by zooming into the feature *edit photo*.



Figure 3.11: Zoom into a feature

Filter

The visualization filters allow users to remove unwanted features from the visualization. We adopt three filters:

- **Filter by quarter:** this filter allows users to focus on a specific period in time, e.g., all the reviews received during one quarter (see Figure 3.12).



Figure 3.12: Filter by quarter

- **Filter by feature volume:** In some cases, the reviews contain numerous features with low volume, which can make the analysis more time consuming. This filter allows users to focus on high-volume features (see Figure 3.13).

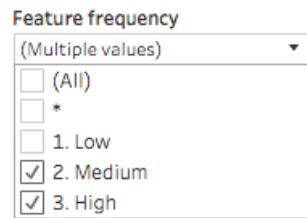


Figure 3.13: Filter by feature frequency

The numeric feature volume is categorized as low, medium or high as following. First, the feature volume range is identified by subtracting the minimum volume from the maximum volume observed in all features from all apps. Then, a feature volume is considered low if it is lower or equal to $1/6$ of the feature range, medium if it is higher than $1/6$ but lower or equal to $2/6$ of the feature range, and high if higher than $2/6$. Figure 3.14 illustrates this concept.



Figure 3.14: Feature frequency categories

- **Filter by SWOT classification:** This filter allows users to focus on features from a certain classification, for example on all strengths. It can also be used to hide features that were not classified under any SWOT quadrant (see Figure 3.15).



Figure 3.15: Filter by SWOT classification

Details-on-demand

This dimension of the task factor concerns the means by which the tool users can obtain details from a group, sub-group or item. We offer two ways to inspect details on demand:

- **Details about a feature:** by hovering over a feature, a tooltip shows more details about it, such as its total frequency and distribution per user rating (see Figure 3.16).

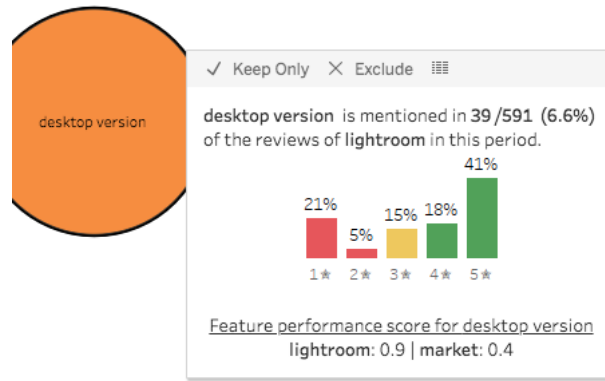


Figure 3.16: Details about a feature

- **Details about a review:** similarly, by hovering over a review, it is possible to read its attributes and inspect the specific sentence where a certain feature is mentioned (see Figure 3.17).

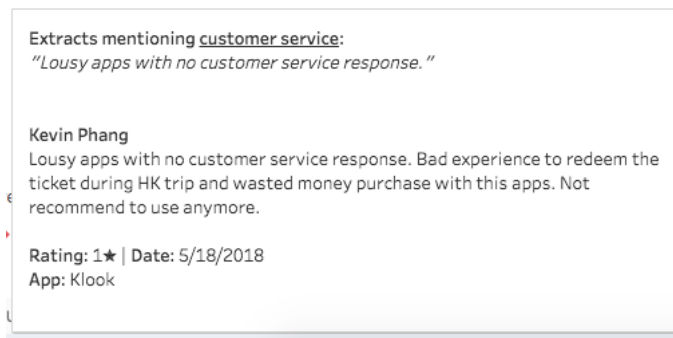


Figure 3.17: Details about a review

Relate

This dimension allows users to explore the relationships in the data. There are two main relationships in the visualized data: mention(feature, review) and SWOT(feature, app). The first one is enabled by clicking on a feature. All the reviews where such feature is mentioned are displayed on the bottom, and their color represents the review sentiment (see Figure 3.18).

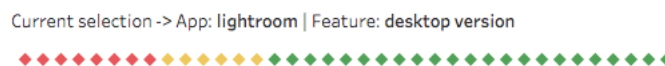


Figure 3.18: Reviews mentioning a feature

The second relationship is visualized through colors. Each feature has one of the following colors: green (strength), red (weakness), blue (opportunity), orange (threat), or grey (other).

History

This dimension describes how users can undo their actions. When performing any action such as zooming into a feature or inspecting their associated reviews, users can restore the previous state by clicking outside the feature. Besides that, any Tableau visualization has three options: undo, redo, or revert (see Figure 3.19).

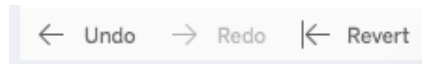


Figure 3.19: History options

Extract

This dimension refers to how users can extract or focus on a sub-collection from a given set. During the NLP phase, we group features that are semantically similar. However, there might be still features that are not synonyms, but refer to the same capability, e.g., “multiple devices” and “account syncing”. In this case, users can remove one of the redundant features from the visualization by clicking on the feature and selecting “Exclude” (see Figure 3.20).

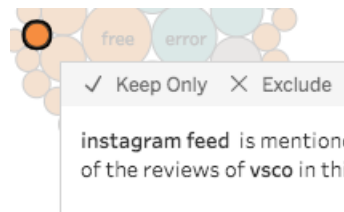


Figure 3.20: Exclude a feature

Chapter 4

RE-SWOT evaluation

In this chapter we discuss the evaluation of the RE-SWOT method and its associated tool through observational case studies with three companies. An observational case study is a study of a real-world case without performing an intervention [84]. In this type of evaluation, generalization occurs by analytical induction over cases, rather than statistical inference from samples [84]. Section 4.1 presents the evaluation protocol used, Section 4.2 describes the three cases, and Section 4.3 presents the results and conclusions obtained from the interviews.

4.1 Evaluation protocol

4.1.1 Goal

The goal of the evaluation can be described as following:

In which ways can RE-SWOT support requirements elicitation through competitor analysis?

To reach this goal, semi-structured interviews were conducted with three practitioners from different industries. In a semi-structured interview, there is an incomplete script that leaves space for improvisation during the interview. The reason for that is to facilitate a deeper understanding of a social situation [62].

4.1.2 Interview protocol

Due to logistic constraints, the interviews were conducted remotely through online meetings that allowed for screen sharing and recording. The interview protocol, which is fully presented at Appendix A, consisted of the following parts.

Introduction (5 min): The participant receives an explanation about the research and the goal of the interview.

Contextual questions (10 min): The participant answers questions about their company, their role, the product and their current practices when analyzing reviews.

Demo (10 min): The participant receives an explanation about the tool principles and how it can be used to do competitive analysis.

Tool usage (20 min): The participant has up to 20 min to freely interact with the tool, while receiving minimum interference from the researcher. The participant is encouraged to think aloud while using the tool, and they can finish whenever they think no more insights can be obtained from interacting with the tool.

Follow-up questions (15 min): The participant answers questions related to their experience with the tool. Such questions are meant to surface any advantages or disadvantages of the tool, points that did not correspond to their needs and expectations, and a comparison to their current practice.

4.1.3 Sampling

For sampling the participants, a call for participating on the research was posted in the online community MindtheProduct [8], which is an international community for product management. Besides that, one of the participating companies was approached directly and accepted to contribute to the evaluation. To be able to participate on the research, candidates had to fit the following selection criteria:

- The company has a mobile app distributed on Google Play or iOS App Store.
- The mobile app and 2 competitors have received at least 200 reviews in English each over the same time period and distribution platform.

4.2 Case descriptions

4.2.1 PlentyOfFish

PlentyOfFish (POF) is a Canadian company founded in 2003 that offers online dating services. The organization has approximately 95 employees and is based on Vancouver, Canada. Since

2015, POF is part of the Match group, which also owns other online dating services such as Tinder, Match.com and OKCupid [68].

POF has 150 million registered users and 65 thousand new daily users. It is available in 11 languages and more than 20 countries through a website and mobile applications for iOS, Android and Windows Phone [68].

POF (Figure 4.1) is free to use, but generates revenue through advertising and premium memberships. POF offers different features to facilitate online meeting, such as creating a personal profile, browsing other users, being *matched* with potential partners and advanced messaging capabilities.

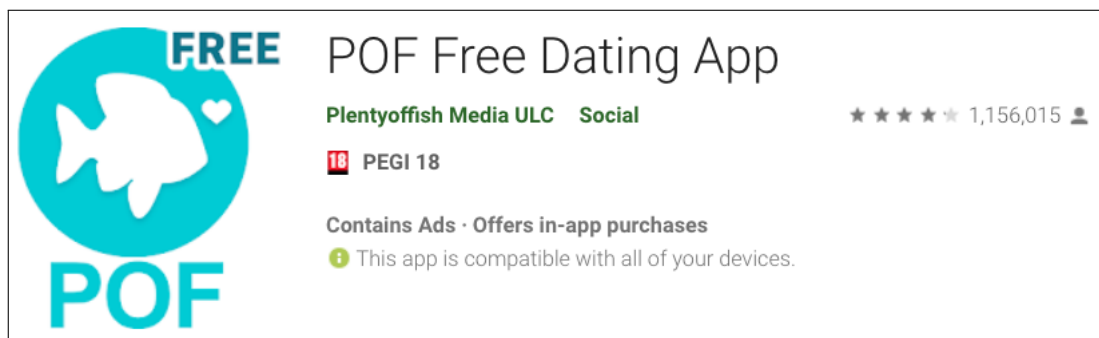


Figure 4.1: POF app on Google Play [7]

The interviewee was a Senior Business Analyst with 2 years of experience in the role and company. Part of their responsibilities include obtaining input from the market, users and internal stakeholders to elicit requirements for POF mobile apps.

The two competitors chosen for analysis by the participant were Badoo and Bumble. Badoo is an online dating service with 390 million registered users and 300 thousand new daily sign ups [4]. On the other hand, Bumble is a location-based dating application with 22 million users which was considered America's fastest-growing dating-app company [63].

In total, 5,014 reviews were collected from POF and its competitors on Google Play. The reviews were posted during May 2018, as detailed on Table 4.1.

4.2.2 GetYourGuide

GetYourGuide is a Switzerland-based company targeted at tourists where users can book activities and buy tickets for attractions in their destination. Its services are available in three platforms: a web application, an Android application [5] and an iOS application [11].

App	Reviews	Features identified
POF	3,220	280
Badoo	992	46
Bumble	802	66

Table 4.1: Data collected for POF (period: from 01/05/2018 to 31/05/2018)

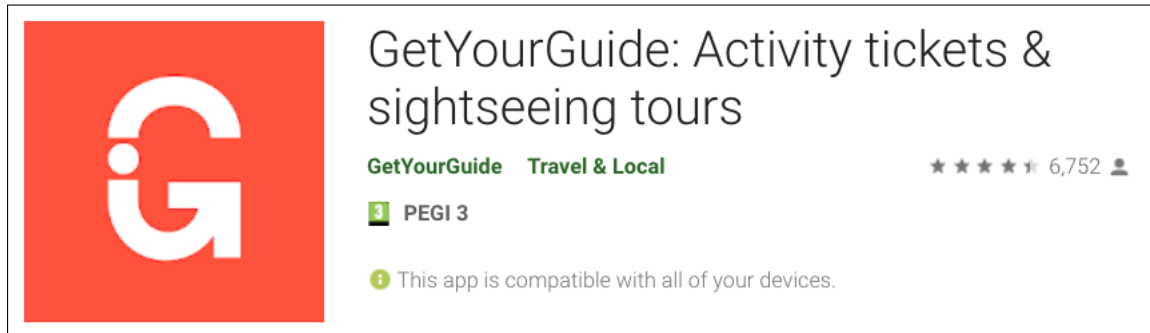


Figure 4.2: GetYourGuide app on Google Play [5]

The company was founded in 2008 and has approximately 400 employees distributed over offices in Europe, North America and the Middle East [11]. GetYourGuide's features include ticket booking, offline availability, and personalized recommendations.

The interviewee was a Senior Product Manager whose responsibilities include overseeing the development roadmap for GetYourGuide mobile apps. The participant had over 6 years of experience in Product Management and has been working for GetYourGuide for approximately one year.

The interviewee named the apps Klook and Viator as competitors for the tool evaluation. Klook is an Asia-focused travel activity platform launched in 2014, while Viator is an app for booking global attractions that is part of TripAdvisor Media Group [28]. Reviews were collected from Google Play for the period from 01/12/2016 to 31/05/2018. Table 4.2 describes the amount of reviews collected for GetyourGuide and how many features were identified.

App	Reviews	Features identified
GetYourGuide	253	9
Klook	506	30
Viator	375	14

Table 4.2: Data collected for GetYourGuide (period: from 01/12/2016 to 31/05/2018)

4.2.3 Blob Connect

Blob Connect (Figure 4.3) is a game developed by AppTornado. In the game, players need to connect dots to collect points and advance levels. It can be played online or offline, and it has 135 puzzles of different types.

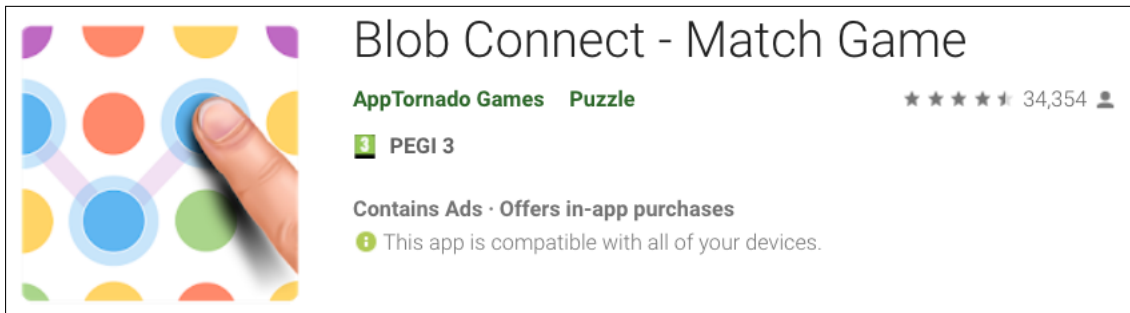


Figure 4.3: Blob Connect app on Google Play [3]

AppTornado is an app development company founded in 2009 by two former Google employees. Under the name Swiss Codemonkeys, the company has published more than 30 Android apps in different categories, summing up to 80 million downloads in total.

The company has 6 employees who are distributed over two offices: one in Zurich, Switzerland and another one in Utrecht, Netherlands. The interviewee was the CEO of the company, who had a background in Software Engineering and 9 years of experience developing apps.

For the evaluation, two competitors were analyzed: Spots Connect and Dots & Co. Similarly to Blob Connect, in both games users need to connect dots to make points. The review data was collected from Google Play for the period from 01/01/2017 to 15/06/2018. Table 4.3 summarizes the data collection for Blob Connect.

App	Reviews	Features identified
Blob Connect	743	11
Spots Connect	2,105	72
Dots & Co	3,321	176

Table 4.3: Data collected for Blob Connect (period: from 01/01/2017 to 15/06/2018)

4.3 Results and conclusions

In this section, we report the results of the interviews conducted with the experts from POF, GetYourGuide and Blob Connect. First, we present the current practice for analyzing app store reviews. Then, the positive aspects and any insights generated while using it are reported. Third, we present the suggestions participants made for improvement and any missing features. Finally, we present their comparison to their current practice, and factors that would influence their adoption of RE-SWOT.

4.3.1 Current practice

All interviewees reported reading their app reviews in some degree. On POF, the interviewee mentioned to read through all reviews approximately once per month to obtain an overview of user perception about the app and to identify areas for improvement. On GetYourGuide, on the other hand, an integration is used to post all reviews into their team communication channel, from where they can be read on an ongoing basis. According to the interviewee from this company, this setup gives the team the ability to react quickly if critical bugs are reported. Finally, the participant from Blob Connect said to occasionally read reviews through Google Play Developers console, but mentioned that they often do not contain relevant information that can lead to changes on product development.

Surprisingly, none of the interviewees reported to have the habit of reading their competitors reviews. For POF, the participant mentioned they would like to analyze competitors reviews more often, but time constraints are the main factor to not do so. On the other hand, participants from GetYourGuide and Blob Connect suggested they never considered competitors reviews as a source of potential requirements.

4.3.2 Positive aspects and insights generated

All participants pointed that the visual and interactive aspect of RE-SWOT is the main positive aspect of the tool. According to POF interviewee, “The tool is easy to learn and navigate”. On GetYourGuide case, the participant described the experience as a “deep-dive analysis”, and reported it as an interesting way to obtain an overview of user reviews about the competition. Similarly, Blob Connect participant said that “*being able to browse the reviews in such a visual way*” was the most distinctive aspect of their experience.

On POF, other positive aspects of RE-SWOT were highlighted. According to the participant,

the ability of going from a high-level view to a more detailed level would positively impact the communication of insights to different stakeholders. Moreover, the interviewee referred to the automated detection of phrases from the reviews as an helpful way to filter a large volume of reviews. Finally, the option to download review data was also perceived by this participant as an advantage.

When asked about insights generated while using the tool, participants gave different answers. For POF, the fact that one of the competitors received less reviews was not expected by the interviewee, and thus represented a new information. For GetYourGuide, the participant stated that they became aware of a feature from the competitors that was previously unknown. When asked about how this could influence product development, they said: *“If I have time, this is something I will investigate better to understand what exactly it is, why people like it and whether this impacts our app”*. Another insight was that one of the competitors had done a promotion: “I could also see that one competitor did some kind of promotion because people were commenting about promo codes. I don’t know if this is something I can act on, but it is good to know.”

Finally, Blob Connect interviewee stated that the tool provided a confirmation that competitor apps adopt a similar business model, but that insight generation was limited, mostly due to the nature of reviews received by their kind of app. *“If I see my competitors have a payment problem, like if users think it is too expensive, I could promote my app to say we are cheaper than them. But on the other hand, if they say they are not happy with the graphics, this is not necessarily an information I can act on.”* According to the interviewee, reviews on the gaming industry tend to be “generic and very little informative”, which tends to hinder knowledge discovery from this source.

4.3.3 Improvements and missing features

All participants named a number of potential improvements or missing features during the interviews.

For POF participant, being able to easily see trends over time, rather than quarterly snapshots of user opinion, would be the most valuable improvement. According to the interviewee, identifying changes or trends on the market would make the tool more actionable. The participant also mentioned as a missing feature the ability to see an overall breakdown of user ratings per app, rather than only per feature.

Similarly, GetYourGuide participant also mentioned that the current visualization does not allow for a temporal view of the features. When asked about how this could support their work, the interviewee said: *“If there is a spike (of volume) in a threat from one quarter to the other, I wonder what did my competitor do that caused this. Perhaps there is a relationship between this feature and something else, which I would need to investigate further”*. They also missed the ability to look at reviews at custom periods of time, rather than only quarter by quarter.

The interviewee also added that knowing for how much time a feature is being negatively mentioned could help understand the priority of addressing the underlying issue. Another topic discussed by GetYourGuide participant was the feature extraction: *“I think being able to merge a lot of these features could be helpful to make them comparable. And if you think about it, many of them are not actually features, but other things that people talk about. So I would have to spend some time thinking on how to take action on these.”*. When it comes to the SWOT classification, the participant stated that in some cases it might be a false positive: *“If they have a negative review coming from a bug, I don’t know how to phrase this as an opportunity to my app. Everyone has problems, but not all problems are opportunities to me”*.

Finally, Blob Connect interviewee mentioned potential improvements that could benefit usability, and make the tool more useful in the long term. The participant discussed ways that the tool could filter or remove uninformative reviews, such as allowing users to focus on reviews that have at least a certain length. Another suggestion was to offer a preview of reviews mentioning a certain feature from the initial view, instead of having to first select the feature and then hover over the reviews. According to the participant, this would make it easier to decide whether to click or not on a feature, and make usage more efficient.

The Blob Connect participant also noted that in most cases different features might refer to the same underlying idea and as such the feature extraction algorithm could be improved, or a way to combine features could be offered. Besides that, they referred to the fact that some opportunities suggested by RE-SWOT were not accurate in a real-world scenario, since they referred to game mechanics that were also used by their own app.

4.3.4 Comparison to current practice

As discussed previously, none of the participants currently read their competitors reviews. Therefore, when comparing RE-SWOT to their current practice, most interviewees focused on

how they currently read their own reviews.

The POF participant mentioned that the feature generation was an improvement to their current practice, since they now search for keywords on their own reviews to understand user opinion about a topic. When asked to compare their experience with their current practice, GetYourGuide interviewee stated *“I have never read my competitors reviews, so using this visualization tool was better than not reading them. I guess the more knowledge, the better”*. Finally, Blob Connect interviewee mentioned that Google Play Console provides review highlights (illustrated on Figure 4.4), which rely on a similar concept as the feature extraction used by RE-SWOT. However, according to the participant, analyzing reviews through RE-SWOT was more visual and interactive.

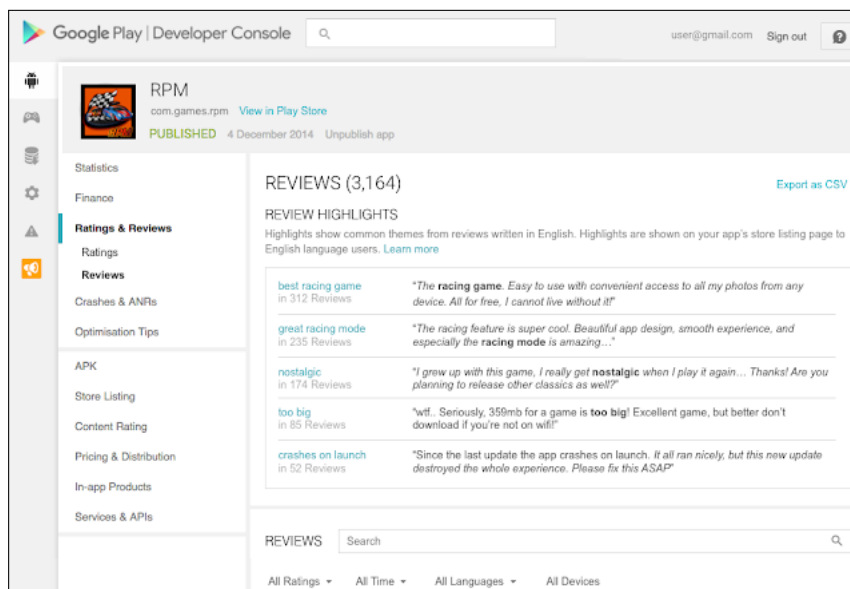


Figure 4.4: Review highlights from Google Play Developers Console[12]

4.3.5 Factors influencing adoption

The last topic discussed on the interviews was how RE-SWOT could be adopted as part of the participants workflow, and any factors that could hinder their adoption.

POF participant stated that they could see RE-SWOT being integrated into their workflow if there would be a way to add more competitors, and if it was possible to see trends and changes over time rather than only quarterly snapshots. When asked about factors that could hinder the adoption of the tool in a real-case scenario, the interviewee mentioned that the fact that the tool currently relies on Tableau Software could make it unaffordable to most companies.

According to GetYourGuide participant, if the tool was improved to show trends, allow the combination of features and let users set custom periods of time, then it would be more useful to be adopted in their workflow. *“We could use it to sit down and re-write some of our content, or check the roadmap to see if there is something we need to change”*. When asked about the frequency with which the tool would be used, the participant stated that with the current quarterly setup, the tool could be used for quarterly or yearly planning, while with custom time filter, then it could be used every week or two weeks to monitor previous issues and discover new ones.

Finally, the Blob Connect interviewee said that currently there would not be a strong incentive to adopt the tool: *“Unless there is some feature that warns me if there is a change, like if there was an upgrade and comments are significantly better now, then I cannot see a benefit on coming back to this tool another time”*. The participant said that due to the volume of reviews their app receives, the current quarterly analysis makes the tool less useful: *“For an app like ours, I feel like most of the changes from quarter to quarter will be random and not worth further attention”*.

4.3.6 Conclusions

In this section, we reflect on the results obtained from the interviews with practitioners from POF, GetYourGuide and Blob Connect. The positive and negative aspects of RE-SWOT are weighted to answer the evaluation goal:

In which ways can RE-SWOT support requirements elicitation through competitor analysis?

The major findings obtained from the evaluation are discussed according the four themes: feature detection, SWOT classification, change detection and visualization.

Feature detection

On RE-SWOT development, NLP techniques are used to automatically detect potential app features from the app store reviews. The automated nature of feature detection has advantages and drawbacks, as the evaluation showed. While feature detection was considered as an advantage by POF participant, the fact that it generates false positives and duplicates had more importance on the feedback of interviewees from GetYourGuide and Blob Connect.

One possible explanation for such difference is the fact that POF receives, on average, more than 100 reviews per day on Google Play, which is approximately 100 times the rate of reviews

received by GetYourGuide and Blob Connect in the same period. With a far superior volume of reviews received, the perceived usefulness of being able to filter reviews by common topics or features might be bigger than the perceived effort of having to disconsider duplicates or remove false positives.

We conclude that the feature detection algorithm does provide value for product managers of apps with a high inflow of reviews. However, improvements are needed to reduce the generation of features that refer to the same underlying issue – for example, “watch ads” and “annoying ads” being detected as separated issues from sentences as “I have to watch annoying ads”. Moreover, not all collocations detected correspond to app features or topics that are relevant for product development, which reinforces the need for a way to remove unnecessary features from the visualization.

SWOT classification

On RE-SWOT, features are classified as strengths, weaknesses, threats or opportunities depending on their performance in the market. The main goal of such classification is to provide product managers a starting point for the identification of new requirements that could improve their app’s competitiveness.

Interviewees demonstrated to understand the concept behind such classification and actively used it to spot positive or negative aspects of their app and their competitors. However, a common feedback about the SWOT classification was that a positive or negative aspect of the competition might not necessarily be a threat or an opportunity to your app. It could be argued that other factors such as company strategy, target audience, and gravity of the issue need to be simultaneously considered to determine what constitutes a real strength, weakness, threat or opportunity to a business or an app.

While the SWOT classification in RE-SWOT was designed to be the main bridge between user feedback and new requirements, the interviews suggested that, in reality, user feedback from the reviews are yet another piece of the puzzle of requirements elicitation. Even if RE-SWOT might generate insights about positive or negative aspects of the apps under analysis, these insights might not be sufficient on their own to derive new requirements.

Change detection

One negative aspect of the tool emphatically mentioned by all participants was that RE-SWOT should support change detection in the reviews in order to be more actionable. As participants

from GetYourGuide and POF reported, being able to see trends over time would be one of the most important functionalities in the tool. Similarly, Blob Connect interviewee explained that being notified about new features or relevant changes in sentiment could trigger new questions or insights that influence product development.

This finding represents a contribution to the literature in the sense that it exposes an important need that practitioners have when eliciting requirements from app store reviews. While previous research solely focused on ways to summarize user feedback present on the reviews, the evaluation suggests that practitioners would also benefit from approaches that incorporate or support anomaly detection.

Visualization

The final major finding from the evaluation relies on the fact that using an interactive visualization to browse the reviews was reported as a positive factor by all practitioners interviewed. Benefits mentioned include being able to easily switch between a high-level view and a detailed inspection of the reviews, facilitating communication among different stakeholders, and easily spotting the main topics mentioned by users. This finding adds to previous research on the use of Information Visualization on Requirements Engineering, and extends its benefits to the field of Crowd-based Requirements Engineering (CrowdRE).

Chapter 5

Discussion

In this research, we studied how Natural Language Processing (NLP) and Information Visualization (IV) can be used to support requirements elicitation from app store reviews. More specifically, we focused on supporting competitive analysis from user reviews as a way to identify improvements or changes to a product.

In this chapter, we provide answers to the research questions presented on Chapter 1, discuss the limitations of the research and propose questions to be addressed by future work on this topic.

5.1 Answering the sub research questions

RQ1: How do practitioners analyze app reviews?

Section 2.2.4 detailed the state of practice in app reviews analysis. The literature revealed that app development companies often employ so-called community managers, who are responsible for manually reading and replying to user reviews. During this process, new issues are eventually identified and raised to the team, leading to new requirements [51].

Besides that, gray literature showed that practitioners might employ commercial tools such as App Annie, Appbot and Appfollow to aggregate reviews from different sources, make them more easily accessible by the team, and filter reviews by metadata (such as rating, time, and app versions). However, the identification of issues that lead to requirements occurs by manually finding common denominators across the reviews.

RQ2: What obstacles do practitioners face when analyzing app reviews?

As presented on Section 2.2.4, main obstacles encountered by practitioners stem from the fact that reviews are noisy, unstructured, and lack context [27] [42] [55]. Therefore, analyzing a high-volume of reviews is time-consuming, tedious, and difficult to scale [42] [51].

Generic classification approaches (which offer ways to tag reviews into a fixed taxonomy) are already available at commercial tools. However, actionable insights are often derived from understanding user opinion about app-specific features [75], which currently cannot be extracted from such tools. Therefore, the challenge faced by practitioners rely on obtaining actionable, app-specific insights in an efficient way.

RQ3: Which applications of NLP already exist in Crowd-based requirements engineering and what can be learnt from them?

As showed on Chapter 2.3, NLP techniques have been widely used in the literature to mine app store reviews and tackle the obstacles listed on RQ2. One type of application widely explored on the literature is the classification of reviews. Past studies [65] [56] [33] proposed approaches to classify reviews according to review topic (e.g. performance, pricing, etc.) and/or user intention, e.g. whether reviews contained bugs, feature requests, or questions.

Another application of NLP concerns the extraction of app-specific features from the reviews, as a way to pinpoint what parts of the app are being mentioned. Studies in this area [44] [43] [50] employ collocation finding algorithms to model features as a group of n words that often appear together. As a general learning, the literature showed that preprocessing steps, such as stopword removal, stemming and lemmatization, should be used to filter out noise and prepare the reviews for further modelling.

Finally, the result of NLP applications often have the form of a summary that can be used by practitioners to identify and prioritize the most important issues for product development. In such summaries, reviews are aggregated according to topics or features, and information that helps derive priority, such as priority scores, sentiment, volume, or frequency, are provided.

RQ4: What types of Requirements Engineering visualizations already exist and what can be learnt from them?

Section 2.4 showed that visualization has been widely applied to Requirements Engineering to support tasks such as detecting ambiguity in requirements [31], understanding risks [35], and navigating interdependencies [72].

Cooper et al. [29] categorized Requirements Engineering visualizations into five types: tabular visualizations, relational visualizations, sequential visualizations, hierarchical visualizations and quantitative visualizations. The latest type helps convey relative data and may use visual metaphors to convey meaning at a glance [29].

Although Requirements Elicitation has become the most supported phase of Requirements Engineering [13] in terms of visualization, applications to distributed requirements elicitation and CrowdRE remain little explored in the literature. The state of the practice, presented on 2.4.3, revealed that existing commercial tools already employ some degree of visualization to support the analysis of app store reviews.

Based on the framework proposed by Reddivari et al. [70], it was showed that the visualizations used by such commercial applications score high on the *user* and *data* factors, but low on the *knowledge* and *visualization* factors. Although they have limited interactivity and are sub-optimal in supporting the knowledge discovery process, these visualizations are intuitive, practitioner-oriented and allow for the processing of large-scale inputs.

RQ5: How to build a tool that uses NLP and IV to help requirements engineers analyse app reviews?

Based on the problem investigation and on how existing applications of IV and NLP support requirements elicitation from app store reviews, our contribution focused on the task of competitive analysis from reviews. For that, a novel approach, called RE-SWOT, was introduced on Chapter 3. Its conceptual framework is derived from SWOT analysis and points an app's strengths, weaknesses, opportunities, and threats. The comparison among all apps is supported by a Feature Performance Score (FPS) which takes into account the sentiment and volume of reviews mentioning a certain feature. Learnings from the literature on NLP are applied to extract app-specific features from the reviews and verify whether the same feature is present across competing apps. By using FPS, the features are then classified as strengths, weaknesses, opportunities or threats.

On the top of that, an interactive visualization is constructed on Tableau Software. Such visualization allows practitioners to go from a high-level view with all features and their SWOT classification into a more detailed view where it is possible to inspect which extracts of the reviews mention each feature, so to investigate why users like or dislike a feature. Filters for feature volume, time, and SWOT classification are provided to allow practitioners to focus on

specific features on time. The prototype is available as a stand-alone tool built on Shiny, where requirements engineers can upload a dataset of reviews, and obtain as output an interactive visualization of the features across all apps.

RQ6: Does the proposed tool satisfy the expectations of practitioners to help identify and prioritize requirements derived from app reviews?

Finally, the evaluation of RE-SWOT is presented on Chapter 4. To answer RQ6, observational case studies were conducted with real practitioners of three companies from different industries. All participants (a Product Manager, a Business Analyst and a CEO) were directly involved in the process of requirements elicitation for their mobile apps.

The results of the interviews revealed that RE-SWOT was able to provide new insights to practitioners who need to analyze a large volume of reviews in an ongoing basis, and that the use of visualization was regarded as a strength of the approach. However, to satisfy the expectations and needs of practitioners, RE-SWOT should additionally provide ways to identify trends and anomalies over time, as reported by all participants. Moreover, as the feature detection algorithm generates false positives, interviewees expected to be able to manually merge or combine features for fine-tuning the analysis.

As the results showed, the insights generated from the reviews are not enough to generate new requirements on its own, but contributes to elicitation by providing information about competitors features and business models, and main points of positive or negative feedback on the apps. These serve as input for discussion by the team, or further investigation.

5.2 Answering the main research question

By answering the sub research questions above, we can now answer the main research question of the project:

MRQ: How to aid requirements engineers analyze app reviews through NLP and Information Visualization?

This question investigates how CrowdRE can benefit from the application of NLP and IV techniques in order to support the task of requirements elicitation from crowd-sourced user feedback, more specifically app store reviews. When evaluating current implementations of software, understanding user opinion about app-specific features is key to make the analysis actionable [75].

In this sense, NLP can aid requirements engineers identify app-specific features or topics from a set of user reviews through collocation finding, i.e. finding groups of words that appear together more often than separately. Such approach has been used on the literature by different studies, and was able to reach average scores of 58% precision and 52% recall [43]. While such scores are far from what can be accomplished by humans, the result of the qualitative evaluation with experts showed that it was able to provide business value for POF, which received approximately 100 reviews per day on Google Play. As the interviews showed, the sub-optimal precision of collocation finding reinforces the need for an interactive tool where practitioners are able to not only eliminate false positives, but also merge *duplicate* features.

It was also showed from the literature that IV can be used on RE to augment a requirements analysts knowledge discovery [70]. The value provided by IV becomes more evident in certain scenarios, including when users are unfamiliar with a collections contents [53].

In order to investigate whether such benefits could also be observed on a CrowdRE setting, an interactive visualization tool was built to allow practitioners navigate by the features and reviews mentioning them. Colors were used to communicate whether the features were strengths, weaknesses, opportunities or threats as per the approach described on Section 3.1. The volume of reviews mentioning each feature was displayed by the size of the circles, and the circle position communicated the app where the feature was observed, and whether it was unique or common across apps. Hovering a feature showed the distribution of user ratings for such feature, and how it compared to the market. Moreover, by focusing on one feature, the visualization also allowed for reading the reviews and the specific extracts mentioning it. Besides filtering capabilities, the tool also let practitioners exclude false positives generated by NLP.

The case studies showed that the interactive visualization was claimed as one of the main advantages of RE-SWOT, with potential benefits such as improving communication, enabling a deep-dive analysis where it is possible to switch between different granularity levels, and providing an engaging analysis experience. However, practitioners missed the ability to see feature performance over time and be able to merge features on the visualization, which should be considered by future works in the area.

5.3 Limitations

The present research has a number of limitations that should be discussed. In this section, we present the research limitations according to four categories: internal validity, conclusion

validity, construct validity and external validity [85].

5.3.1 Internal validity

When investigating whether one factor affects an investigated factor there is a risk that the investigated factor is also affected by a third factor [85]. An important internal validity of this research is the fact that in all case studies, the interviewees currently did not read their competitor reviews. The reason for that is that during the sampling process, the main focus was to recruit participants that were directly involved in the elicitation of requirements for a mobile app, and whose app (and competitors) received a minimum number of reviews that would enable the analysis. Therefore, there is a risk that insights obtained by practitioners about their competitors while using the tool were related to the fact that it was the first time they read competitors reviews, and not driven by the tool itself.

Finally, in order to keep the evaluation context as close to a real scenario as possible, real review data was used on the tool. This resulted in a large variability in terms of volume, time period analyzed and the content of the reviews themselves. These factors are tied to the different context of each app, and might also influence the experience practitioners had with the tool. For instance, the degree to which users refer to actual app features might vary per industry, and represent a threat to internal validity.

5.3.2 Conclusion validity

The conclusion validity is concerned with the relationship between the treatment and the outcome of the experiment [85]. As the evaluation consisted of a qualitative research, a potential threat to conclusion validity is that interviews were coded into observations and therefore there is a risk that the researcher inserts beliefs about the topic during this process.

5.3.3 Construct validity

This validity is concerned with the relation between theory and observation [85]. A threat to construct validity is the fact that RE-SWOT has been evaluated as a whole rather than by its parts. Therefore, it is not fully clear whether the conclusions apply to all parts of RE-SWOT or just some of them. For instance, it is not clear whether the SWOT classification mechanism (which draws a comparison of a feature among all apps to classify it as a strength, weakness, threat or opportunity) is perceived as useful on its own. It could be that practitioners were simply interested in understanding which features had more positive or negative reviews, rather

than knowing if they received more positive reviews than their competitors.

5.3.4 External validity

External validity refers to what extent it is possible to generalize the findings, and to what extent the findings are of interest to other people outside the investigated case [85]. As observational case studies were applied, there is no population from which a statistically representative sample has been obtained. Therefore, generalization is analytical rather than statistical.

Although the participants were real practitioners directly involved with requirements elicitation for mobile apps, one should notice that RE-SWOT is designed for cases where apps receive so many reviews that reading and analyzing them manually represents a significant challenge due to manual effort. In this sense, the application of RE-SWOT to GetYourGuide and Blob Connect cases is not fully representative of the scenario it was intended to, since both apps received less than 10 reviews per day. While their competitors received a higher volume of reviews that would justify the use of automated approaches, this represents a threat to generalizing the results to other cases.

5.4 Future work

The present research offers numerous opportunities for future work in the use of IV and NLP for requirements elicitation. The first possible work would be to investigate how the learnings obtained from the evaluation could be used to adjust the artifact so to better satisfy practitioners needs. Challenges rely on how to facilitate automated anomaly detection of changes on user feedback regarding a specific feature, or detecting when new features start to be mentioned in the reviews in a real-time fashion.

Another future research could focus on how RE-SWOT could be adapted to generalize to different scenarios in CrowdRE. Possible examples would be doing competitive analysis from social media posts, user forums, or other review portals. A generic approach should be able to derive a fine-grained sentiment score from the text rather than from rating taxonomies, and account for different lengths of text. As user feedback for distributed software products can be found in a variety of sources, eliciting requirements from combined sources could potentially give more perspectives or insights that lead to new requirements.

Finally, the present research evaluated RE-SWOT through three case studies. To obtain a broader perspective on the value of the approach, one could investigate its application in several

cases over a period of time. This research could lead to important contributions on how the value of RE-SWOT varies according to contextual factors such as industry, review volume, business models, software development practices, and so on.

Appendix A

Interview protocol

Introduction (5 min)

First of all, thank you for your time and interest in participating in this interview. The main goal of this interview is to obtain a deep understanding about how the tool I developed for my thesis could support you in identifying improvements or changes to your mobile app, based on the competitive analysis from app store reviews.

To facilitate my note-taking, I would like to ask if I can record this conversation today. This recording will not be available to anyone else – this is just for my own use when analyzing the interview results for the research.

This interview consists of three parts:

- First, I will start by making questions related to your organization, your product and how you currently analyze the reviews. The main goal of this phase is to understand your context and your needs.
- Second, I am going to give a demo of the tool so you can understand its basic principles and how it works. After the demo, you will have the opportunity to interact with the tool yourself, so you can explore which kinds of insights you can get from it. During this phase, I will try not to interfere but you can of course make questions if something is not clear to you.
- In the third and last part, we are going to discuss your experience with the tool. Im going to make questions about your perception, and things you see as useful or not useful. There

are no right or wrong answers to these questions, since the main goal is to understand the experience you had.

Do you have any questions so far?

Context (10 min)

- Can you elaborate about what is your role in the organization and how it relates to product development?
- How long have you been working in this role?
- How long have you been working in this organization?
- Can you briefly describe your organization?
- Can you talk about the product you are going to analyze?
- Do you currently read and/or analyze the reviews of your product? If so, can you elaborate on how you do that?
- Do you currently read and/or analyze your competitors reviews? If so, can you elaborate on how you do that?

Demo (10 min)

I am now going to explain the basic principles of the tool I developed. It analyses automatically the reviews from your app and your competitors to identify common features or topics mentioned by users. Then, it compares user opinion about a feature across all apps to classify this feature as a strength, a weakness, a threat or an opportunity to your app. This process is automated, so to minimize your effort. This classification is inspired by the SWOT analysis, which maybe you are familiar with.

- Strength: These are your strong points, i.e. features whose sentiment is positive and above market average.
- Weakness: These are your weak points, i.e. features whose sentiment is negative and below market average.

- **Opportunity:** These are weak points from the competition, i.e. features whose sentiment is negative and below market average.
- **Threat:** These are strong points from the competition that could threaten your app or business, i.e. features whose sentiment is positive and above market average.

The main goal of this tool is to allow you to understand where you stand in the market from a user perspective, and use this information to plan changes to your product. You can think of requirements that will use your strengths and overcome your weaknesses to minimize threats and pursue opportunities.

I am now going to open the tool and show you how it works in practice.

Demonstration of the tool

Do you have any questions about how the tool works?

Tool usage (20 min)

I have prepared the tool with reviews data from your app and 2 competitors, for a given period. I am now going to give you control over the tool so you can use it yourself for as long as you want. Feel free to explore whatever you think is interesting or might give you insights that could influence the future of your app. I will try not to interfere while you are using it, so you can use it as you would normally do in your work.

I would like to ask you to try to think aloud as much as possible, so I can understand what you are thinking and/or trying to achieve. If you have any further questions about how the tool works, please feel free to ask them as well.

Give interviewee control over the tool

Follow-up questions (15 min)

- How would you describe your experience with this tool?
- Can you give examples of insights you had while using the tool, and how they could influence product development?
- Could you elaborate on aspects of the tool that you think are the most useful?

- Could you elaborate on how this tool could be improved?
- How would you compare this tool with your current practice for analyzing reviews from your app and/or competitors?
- How do you think this tool could be integrated into your current workflow?
- Do you see any limitations on this tool that could hinder its adoption by practitioners?

Do you have any other remarks or comments to make?

We have reached the end of this interview. I would like to thank you for your participation and feedback, as it was very important for this research.

Bibliography

- [1] App review & ratings analysis for mobile teams. <https://appbot.co/>.
- [2] Appanalytics — mobile app analytics. <http://appanalytics.io/>.
- [3] Blob connect - match game - apps on google play.
- [4] Discover badoo.
- [5] Getyourguide: Activity tickets sightseeing tours - apps on google play.
- [6] Mobile customer experience software for the fortune 1000. <https://www.apptentive.com/>.
- [7] Pof free dating app - apps on google play.
- [8] Product management slack community.
- [9] Roadmap prioritization from product feedback. <https://www.uservoice.com/>.
- [10] Tableau software. <https://www.tableau.com/>.
- [11] We turn trips into amazing experiences.
- [12] New tools for ratings reviews on google play to engage and understand your users, Feb 2016.
- [13] Zahra Shakeri Hossein Abad, Mohammad Noaen, and Guenther Ruhe. Requirements Engineering Visualization: A Systematic Literature Review. *2016 IEEE 24th International Requirements Engineering Conference (RE)*, (September):6–15, 2016.
- [14] Raian Ali, Carlos Solis, Mazeiar Salehie, Inah Omoronyia, Bashar Nuseibeh, and Walid Maalej. Social sensing: when users become monitors. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pages 476–479. ACM, 2011.

- [15] App Annie. App annie - the app analytics and app data industry standard. <https://www.appannie.com>.
- [16] AppFollow. Reviews & updates monitor for app store & google play. <https://appfollow.io/>.
- [17] Apptentive. <https://www.apptentive.com/resource/2016-guide-mobile-app-ratings-reviews>.
- [18] Joy Beatty and Mike Alexander. Display-action-response model for user interface requirements: Case study. In *Requirements Engineering Visualization, 2007. REV 2007. Second International Workshop on*, pages 10–10. IEEE, 2007.
- [19] Kenneth Benoit. *quanteda: Quantitative Analysis of Textual Data*, 2018. R package version 0.99.22.
- [20] Daniel Berry. Natural language and requirements engineering-nu. In *International Workshop on Requirements Engineering, Imperial College, London, UK*, 2001.
- [21] Kamaljit Kaur Bimrah, Haralambos Mouratidis, and David Preston. Modelling trust requirements by means of a visualization language. In *Requirements Engineering Visualization, 2008. REV'08.*, pages 26–30. IEEE, 2008.
- [22] Pierre Bourque, Richard E Fairley, et al. *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press, 2014.
- [23] Remo Burkhard. Towards a framework and a model for knowledge visualization: Synergies between information and knowledge visualization. *Knowledge and information visualization*, pages 29–42, 2005.
- [24] Claire Capon. *Understanding organisational context: Inside and outside organisations*. Pearson Education, 2004.
- [25] Stuart K Card, Jock D Mackinlay, and Ben Shneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999.
- [26] Rishi Chandy and Haijie Gu. Identifying spam in the ios app store. In *Proceedings of the 2nd Joint WICOW/AIRWeb Workshop on Web Quality*, pages 56–59. ACM, 2012.
- [27] Ning Chen, Jialiu Lin, Steven C. H. Hoi, Xiaokui Xiao, and Boshen Zhang. AR-miner: mining informative reviews for developers from mobile app marketplace. *Proceedings of*

- the 36th International Conference on Software Engineering - ICSE 2014*, pages 767–778, 2014.
- [28] Gobinda G Chowdhury. Natural language processing. *Annual review of information science and technology*, 37(1):51–89, 2003.
- [29] John R. Cooper, Seok Won Lee, Robin A. Gandhi, and Orlena Gotel. Requirements engineering visualization: A survey on the state-of-the-art. *2009 4th International Workshop on Requirements Engineering Visualization, REV 2009*, pages 46–55, 2009.
- [30] Cortical.IO. Cortical.io. <http://api.cortical.io/>.
- [31] Fabiano Dalpiaz, Ivor van der Schalk, and Garm Lucassen. Pinpointing ambiguity and incompleteness in requirements engineering via information visualization and nlp. REFSQ 2018: 24rd International Working Conference on Requirements Engineering: Foundation for Software Quality, 2018.
- [32] Maya Daneva. What do we know about the actual use of crowdsourced feedback in support of re activities? UU/SIKS Symposium on Natural Language in Requirements Engineering, Utrecht, 2017.
- [33] Andrea Di Sorbo, Sebastiano Panichella, Carol V. Alexandru, Junji Shimagaki, Corrado A. Visaggio, Gerardo Canfora, and Harald C. Gall. What would users change in my app? summarizing app reviews for recommending software changes. *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2016*, pages 499–510, 2016.
- [34] Davide Falessi, Giovanni Cantone, and Gerardo Canfora. Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques. *IEEE Transactions on Software Engineering*, 39(1):18–44, 2013.
- [35] Martin S Feather, Steven L Cornford, James D Kiper, and Tim Menzies. Experiences using visualization techniques to present requirements, risks to them, and options for risk mitigation. In *Requirements Engineering Visualization, 2006. REV'06. First International Workshop on*, pages 10–10. IEEE, 2006.
- [36] Ingo Feinerer and Kurt Hornik. *tm: Text Mining Package*, 2017. R package version 0.7-1.

- [37] J. D. Fekete, J. V. Wijk, J. Stasko, and C. North. The Value of Information Visualization To cite this version : The Value of Information Visualization. *Lecture Notes in Computer Science*, pages 1–18, 2008.
- [38] Bin Fu, Jialiu Lin, Lei Li, Christos Faloutsos, Jason Hong, and Norman Sadeh. Why people hate your app: Making sense of user feedback in a mobile app store. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1276–1284. ACM, 2013.
- [39] Cuiyun Gao, Baoxiang Wang, Pinjia He, Jieming Zhu, Yangfan Zhou, and Michael R. Lyu. PAID: Prioritizing app issues for developers by tracking user reviews over versions. *2015 IEEE 26th International Symposium on Software Reliability Engineering, ISSRE 2015*, pages 35–45, 2016.
- [40] Necmiye Genc-Nayebi and Alain Abran. A systematic literature review: Opinion mining studies from mobile app store user reviews. *Journal of Systems and Software*, 125(November):207–219, 2017.
- [41] Orlena CZ Gotel, Francis T Marchese, and Stephen J Morris. The potential for synergy between information visualization and software engineering visualization. In *Information Visualisation, 2008. IV'08. 12th International Conference*, pages 547–552. IEEE, 2008.
- [42] Eduard C. Groen, Norbert Seyff, Raian Ali, Fabiano Dalpiaz, Joerg Doerr, Emitza Guzman, Mahmood Hosseini, Jordi Marco, Marc Oriol, Anna Perini, and Melanie Stade. The Crowd in Requirements Engineering: The Landscape and Challenges. *IEEE Software*, 34(2):44–52, 2017.
- [43] Emitza Guzman and Walid Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, pages 153–162. IEEE, 2014.
- [44] Mark Harman, Yue Jia, and Yuanyuan Zhang. App store mining and analysis: MSR for app stores. *IEEE International Working Conference on Mining Software Repositories*, pages 108–111, 2012.
- [45] Jonathan Hey. The data, information, knowledge, wisdom chain: the metaphorical link. *Intergovernmental Oceanographic Commission*, 26, 2004.

- [46] Mahmood Hosseini, Keith Phalp, Jacqui Taylor, and Raian Ali. Towards crowdsourcing for requirements engineering. *CEUR Workshop Proceedings*, 1138:82–87, 2014.
- [47] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '04*, page 168, 2004.
- [48] Slinger Jansen and Ewoud Bloemendal. Defining app stores: The role of curated marketplaces in software ecosystems. In *International Conference of Software Business*, pages 195–206. Springer, 2013.
- [49] Jian Jin, Ping Ji, and Rui Gu. Identifying comparative customer requirements from product online reviews for competitor analysis. *Engineering Applications of Artificial Intelligence*, 49:61–73, 2016.
- [50] Timo Johann and Walid Maalej. Democratic mass participation of users in Requirements Engineering? *2015 IEEE 23rd International Requirements Engineering Conference, RE 2015 - Proceedings*, pages 256–261, 2015.
- [51] Swetha Keertipati, Bastin Tony Roy Savarimuthu, and Sherlock A. Licorish. Approaches for prioritizing feature improvements extracted from app reviews. *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering - EASE '16*, pages 1–6, 2016.
- [52] Kurt Koffka. Principles of gestalt psychology, international library of psychology, philosophy and scientific method, 1935.
- [53] Xia Lin. Map displays for information retrieval. *JASIS*, 48(1):40–54, 1997.
- [54] Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.
- [55] Walid Maalej, Hans-Jörg Happel, and Asarnusch Rashid. When Users Become Collaborators: Towards Continuous and Context-Aware User Input. *OOPSLA '09: Proceeding of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications*, pages 981–990, 2009.
- [56] Walid Maalej and Hadeer Nabil. Bug report, feature request, or simply praise? on automatically classifying app reviews. In *Requirements Engineering Conference (RE), 2015 IEEE 23rd International*, pages 116–125. IEEE, 2015.

- [57] Walid Maalej, Maleknaz Nayebi, Timo Johann, and Gunther Ruhe. Towards Data - Driven Requirements Engineering. 33:1–6, 2015.
- [58] Stuart McIlroy, Nasir Ali, Hammad Khalid, and Ahmed E. Hassan. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empirical Software Engineering*, 21(3):1067–1106, 2016.
- [59] Hendrik Meth, Manuel Brhel, and Alexander Maedche. The state of the art in automated requirements elicitation. *Information and Software Technology*, 55(10):1695–1709, 2013.
- [60] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
- [61] Mixpanel. Mixpanel — product analytics for mobile, web, and beyond. <https://mixpanel.com/>.
- [62] Michael D Myers and Michael Newman. The qualitative interview in is research: Examining the craft. *Information and organization*, 17(1):2–26, 2007.
- [63] Clare O’Connor. Billion-dollar bumble: How whitney wolfe herd built america’s fastest-growing dating app, Nov 2017.
- [64] Dennis Pagano and Walid Maalej. User Feedback in the AppStore: An Empirical Study (submitted). *RE ’13: Proceedings of the 21st International Requirements Engineering Conference*, pages 125–134, 2013.
- [65] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado A. Visaggio, Gerardo Canfora, and Harald C. Gall. How can i improve my app? Classifying user reviews for software maintenance and evolution. *2015 IEEE 31st International Conference on Software Maintenance and Evolution, ICSME 2015 - Proceedings*, (1):281–290, 2015.
- [66] Darius Pfitzner, Vaughan Hobbs, and David Powers. A Unified Taxonomic Framework for Information Visualization. 1, 2001.
- [67] Mario Pichler and Hildegard Rumetshofer. Business process-based requirements modeling and management. In *Requirements Engineering Visualization, 2006. REV’06. First International Workshop on*, pages 6–6. IEEE, 2006.
- [68] POF. About plenty of fish. <https://www.pof.com/aboutus.aspx>.

- [69] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.
- [70] Sandeep Reddivari, Shirin Rad, Tanmay Bhowmik, Nisreen Cain, and Nan Niu. Visual requirements analytics: A framework and case study. *Requirements Engineering*, 19(3):257–279, 2014.
- [71] RStudio, Inc. *Easy web applications in R.*, 2013. URL: <http://www.rstudio.com/shiny/>.
- [72] David Sellier and Mike Mannion. Visualising product line requirement selection decision inter-dependencies. In *Requirements Engineering Visualization, 2007. REV 2007. Second International Workshop on*, pages 7–7. IEEE, 2007.
- [73] AM Sen and SK Jain. A visualization technique for agent based goal refinement to elicit soft goals in goal oriented requirements engineering. In *Requirements Engineering Visualization, 2007. REV 2007. Second International Workshop on*, pages 2–2. IEEE, 2007.
- [74] Julia Silge and David Robinson. tidytext: Text mining and analysis using tidy data principles in r. *JOSS*, 1(3), 2016.
- [75] Piotr Spiewanowski. How to visualize and analyze customer feedback, Feb 2016.
- [76] Pratyoush K. Srivastava and Richa Sharma. Crowdsourcing to elicit requirements for My-ERP application. *1st International Workshop on Crowd-Based Requirements Engineering, CrowdRE 2015 - Proceedings*, pages 31–35, 2015.
- [77] Harold D Stolovitch and Erica J Keeps. *Handbook of human performance technology: Principles, practices, and potential*. John Wiley & Sons, 2006.
- [78] Milan Straka and Jana Straková. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, 2017.
- [79] Shiliang Sun, Chen Luo, and Junyu Chen. A review of natural language processing techniques for opinion mining systems. *Information Fusion*, 36:10–25, 2017.
- [80] Anne Treisman. Preattentive processing in vision. *Computer vision, graphics, and image processing*, 31(2):156–177, 1985.
- [81] Usabilla. Usabilla: The standard in user feedback. <https://usabilla.com/>.

- [82] Colin Ware. *Information visualization: perception for design*. Elsevier, 2012.
- [83] Heinz Weihrich. The tows matrixa tool for situational analysis. *Long range planning*, 15(2):54–66, 1982.
- [84] Roel J. Wieringa. *Design science methodology for information systems and software engineering*. Springer, 2014.
- [85] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.
- [86] Hui Yang and Peng Liang. Identification and Classification of Requirements from App User Reviews. *27th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pages 7–12, 2015.
- [87] Didar Zowghi and Chad Coulin. Requirements elicitation: A survey of techniques, approaches, and tools. In *Engineering and managing software requirements*, pages 19–46. Springer, 2005.