Tampere University

XIAOZHOU LI

# Data-Driven Analysis towards Monitoring Software Evolution by Continuously Understanding Changes in Users' Needs

Tampere University Dissertations 556

Tampere University Dissertations 556

XIAOZHOU LI

# Data-Driven Analysis towards Monitoring Software Evolution by Continuously Understanding Changes in Users' Needs

ACADEMIC DISSERTATION
To be presented, with the permission of
the Faculty of Information Technology and Communication Sciences
of Tampere University,
for public discussion in the Room D10b
of Päätalo, Kanslerinrinne 1, Tampere,
on 17 February 2022, at 12 o’clock.

ACADEMIC DISSERTATION
Tampere University, Faculty of Information Technology and Communication Sciences
Finland

| | | |
|---|---|---|
| *Responsible supervisor and Custos* | Adjunct Professor Zheying Zhang<br>Tampere University<br>Finland | |
| *Supervisor* | Associate Professor Konstantinos Stefanidis<br>Tampere University<br>Finland | |
| *Pre-examiners* | Professor Yinglin Wang<br>Shanghai University of Finance and Economics<br>China | Professor Apostolos Zarras<br>University of Ioannina<br>Greece |
| *Opponent* | Professor Filippo Lanubile<br>University of Bari<br>Italy | |

The originality of this thesis has been checked using the Turnitin Originality Check service.



Cover design: Roihu Inc.

ISBN 978-952-03-2300-4 (print)
ISBN 978-952-03-2301-1 (pdf)
ISSN 2489-9860 (print)
ISSN 2490-0028 (pdf)
http://urn.fi/URN:ISBN:978-952-03-2301-1

PunaMusta Oy – Yliopistopaino
Joensuu 2022

*For my mom,* 周莉.
*I love you and miss you a lot.*

# PREFACE/ACKNOWLEDGEMENTS

First of all, I want to thank my two supervisors, Dr. Zheying Zhang and Professor Konstantinos Stefanidis, who have been providing valuable support and guidance during my doctoral studies and leading me through the process. In addition, I am thankful for Professor Yinglin Wang and Professor Apostolos Zarras being the pre-examiners of this thesis and Professor Filippo Lanubile being the opponent.

Specially, I would also like to thank sincerely Professor Davide Taibi and Dr. Valentina Lenarduzzi for their priceless help in the final stage of my academic career. Thank to them and the CloudSEA group, not only had I learnt a lot in academics, but also have I been feeling warmly welcomed to the family. Being my role models, friends and family, they gave me the strength to go through some tough times.

My gratitude also goes to Professor Jyrki Nummenmaa, Biswa Upreti, Dr. Timo T. Poranen, Professor Jaakko Peltonen, Chien Lu, Dr. Timo Nummenmaa, Maria Stratigi, Dr. Boyang Zhang, and Sergio Moreschini (ordered by publication timeline) for being the co-authors through my path of publications. Each experience is unique, memorable, and precious to me in various ways. Special thanks to Nyyti Saarimäki for the help with the Finnish abstract.

Finally, I would love to thank my parents, who have always been supportive to my career with unconditional patience and love.

# ABSTRACT

Software products, though always being expected to provide satisfactory functionalities and be bug-free, somehow fail to meet the expectations of their users. Thus, software maintenance is inevitable and critical for any software companies who want their products or services to continue profiting. On the other hand, due to the fierce competitiveness in the contemporary software market, as well as the ease of user churns, monitoring and sustaining the satisfaction of the users is a critical criterion for the long-term success of any software products within their evolution stage. To such an end, continuously understanding and meeting the users' needs and expectations is the key, as it is more efficient and effective to allocate maintenance effort accordingly to address the issues raised by users.

On the other hand, accompanied by the rapid development of internet technologies, the volume of user-generated content has been increasing exponentially. Among such user-generated content, feedback from the customers, either numeric rating, recommendation, or textual reviews, have been playing an increasingly critical role in product designs in terms of understanding customers' needs. Especially for software products that require constant maintenance and are continuously evolving, understanding of users' needs and complaints, as well as the changes in their opinions through time, is of great importance. Additionally, supported by the advance of data mining and machine learning techniques, the effort of knowledge discovery from analyzing such data and specially understanding the behavior of the users shall be largely reduced.

However, though many studies propose data-driven approaches for feedback analysis, the ones specifically on applying such methods supporting software maintenance and evolution are limited. Many studies focus on the text mining perspective of review analysis towards eliciting users' opinions. Many others focus on the detection and classification of feedback types, e.g., feature requests, bug reports, and emotion expression, etc. For the purpose of enhancing the effectiveness in soft-

ware maintenance and evolution practice, an effective approach on the software's perceived user experience and the monitoring of its changes during evolution is required.

To support the practice of software maintenance and evolution targeting enhancing user satisfaction, we propose a data-driven user review analysis approach. The contribution of this research aims to answer the following research questions: RQ1. How to analyze users' collective expectation and perceived quality in use with data-driven approaches by exploiting sentiment and topics? RQ2. How to monitor user satisfaction over software updates during software evolution using reviews' topics and sentiments? RQ3. How to analyze users' profiles, software types and situational contexts as contexts of use that supports the analysis of user satisfaction? Towards answering RQ1, the thesis proposes a data-driven approach of user perceived quality evaluation and users' needs extraction via sentiment analysis and topic modeling on large volume of user review data. Based on such outcome, the answer to RQ2 encompasses of 1) the approach to monitor user opinion changes through software evolution by detecting similar topic pairs and 2) the approach to identify the problematic updates based on anomalies in review sentiment distribution. Towards the answer to RQ3, a three-fold analysis is proposed: 1) situational contexts and ways of interaction analysis, 2) user profile and preference analysis and 3) software type and related features analysis. All the above approaches are validated by case studies.

This thesis contributes to the examination of applying data-driven end user review analysis methods supporting software maintenance and evolution. The main implication is to enrich the existing domain knowledge of software maintenance and evolution in terms of taking advantage of the collective intelligence of end users. In addition, it conveys unique contribution to the research on software evolution contexts in terms of various meaningful aspects and leads to a potential interdisciplinary contribution as well. On the other hand, this thesis also contributes to software maintenance and evolution practice even in the larger scope of the software industry by proposing an effective series of approaches that address critical issues within. It helps the developers ease their effort in release planning and other decision-making activities.

# TIIVISTELMÄ

Ohjelmistot eivät usein vastaa käyttäjiensä odotuksia siitä huolimatta, että niiden odotetaan tarjoavan riittävä toiminnallisuus ja olevan virheettömiä. Tästä syystä ohjelmiston ylläpito on väistämätöntä ja tärkeää jokaiselle ohjelmistoyritykselle, joka haluaa pitää tuotteensa tai palvelunsa kannattavana. Koska kilpailu nykyajan ohjelmistomarkkinoilla on tiukkaa ja käyttäjien on helppo lopettaa tuotteen käyttö, yritysten on erityisen tärkeää tarkkailla ja ylläpitää käyttäjätyytyväisyyttä pitkäaikaisen menestyksen turvaamiseksi. Tämän saavuttamiseksi tärkeää on jatkuvasti ymmärtää ja kohdata käyttäjien tarpeet ja odotukset, sillä on tehokkaampaa kohdentaa ylläpito käyttäjien esittämien ongelmien perusteella.

Toisaalta internet-teknologiat ovat kehittyneet nopeasti samalla, kun käyttäjien luoman sisällön määrä on kasvanut räjähdysmäisesti. Käyttäjien antama palaute (numeerinen arvostelu, ehdotus tai tekstuaalinen arvio) on esimerkki tälläisestä käyttäjien luomasta sisällöstä ja sen merkitys tuotteiden kehittämisessä asiakkaiden tarpeiden pohjalta kasvaa jatkuvasti. Käyttäjien tarpeiden ymmärtäminen on erityisen tärkeää jatkuvaa ylläpitoa ja kehitystystä vaativissa ohjelmistoissa. Tällöin on myös oleellista ymmärtää, miten asiakkaiden mielipiteet muuttuvat ajan kuluessa. Tämän lisäksi datan louhimisen ja koneoppimisen kehitys vähentävät vaivaa, joka käyttäjän tuottaman datan analysointiin ja erityisesti heidän käyttymisensä ymmärtämiseen tarvitaan.

Vaikka useat tutkimukset ehdottavat tietokeskeistä lähestymistä palautteen arviointiin, ohjelmiston ylläpitoa ja kehitystä hyödyntäviä lähestymistapoja on vähän. Monet menetelmät keskittyvät arvostelujen analysoinnissa tekstinlouhintaan paljastaakseen käyttäjien mielipiteet. Useat menetelmät keskittyvät myös tunnistamaan ja luokittelemaan palautetyyppejä kuten ominaisuuspyyntöjä, virheilmoituksia ja tunteenilmauksia. Jotta ohjelmiston ylläpidosta saataisiin tehokkaampaa, tarvitaankin tehokas lähestymistapa ohjelmiston havaitun käyttäjäkokemuksen ja sen muutosten tarkkailuun ohjelmiston kehittyessä.

Ehdotamme tietokeskeisen käyttäjäarviointien analyysin käyttämistä ohjelmiston ylläpidon ja kehityksen tukemiseksi sekä käyttäjäkokemuksen parantamiseksi. Tämä tutkimus pyrkii vastaamaan seuraaviin tutkimuskysymyksiin: 1) Kuinka analysoida käyttäjien yhteisiä odotuksia ja havainnoitua laatua tunteita ja teemoja hyödyntävällä tietokeskeisellä mallilla? 2) Kuinka voidaan tarkkailla käyttäjien tyytyväisyyttä ohjelmistopäivityksiin arvostelujen teemoja ja tunteita hyödyntäen? 3) Kuinka analysoida käyttäjien profiileita, ohjelmistotyyppejä ja tilannekohtaisia kontekseja niin että ne tukevat aiemmin mainittuja aktiviteetteja? Vastatakseen ensimmäiseen tutkimuskysymykseen väitöskirja ehdottaa tietokeskeistä lähestymistä käyttäjän kokeman laadun arviointiin ja käyttäjän tarpeiden erottamiseen soveltamalla tunneanalyysia ja aihemallinnusta suureen määrään käyttäjien arviointeja. Toisen tutkimuskysymyksen vastaukset sisältävät 1) lähestymistapa käyttäjämielipiteiden muutoksen monitorointiin hyödyntäen ohjelmiston kehitystä samanlaisia aihepareja tunnistamalla 2) lähestymistapa ongelmallisten päivitysten tunnistamiseen käyttäjäarvioista havaittujen tunteiden jakauman poikkeamien peruteella. Vastaus kolmanteen tutkimuskysymykseen on kolmijakoinen 1) tilannekohtaiset kontekstit ja tavat vuorovaikutusanalyysiin 2) käyttäjä profiilien ja mieltymysten analysointi 3) ohjelmistotyyppien ja niihin liittyvien piirteiden analyysi. Kaikki yllämainitut lähestymistavat on validoitu tapaustutkimuksilla.

Väitöskirja edistää ohjelmiston ylläpitoa ja kehitystä hyödyntävien tietokeskeisten loppukäyttäjien arvioiden analyysimetodien tutkimusta. Tärkein tulos on olemassa olevan ohjelmiston ylläpidon ja kehityksen erityistietämyksen kasvattaminen käyttämällä loppukäyttäjien joukkoälyä. Lisäksi esittämällä useita merkittäviä näkökulmia väitöskirja tuo ainutlaatuisen lisän ohjelmiston kehitykseen liittyvään tutkimukseen ja johtolankoja mahdolliseen alojen väliseen yhteistyöhön. Toistaalta väitöskirja edistää ohjelmiston ylläpitoa ja kehitykstä ohjelmistoalalla myös laajemmalti sillä se esittää tehokkaan joukon lähestymistapoja, jotka tuovat sisältäpäin esiin vakavia ongelmia. Tämä auttaa kehittäjiä vähentämään vaivaa julkaisun suunnittelussa ja muissa päätöksentekoa vaativissa tehtävissä.

# CONTENTS


1 Introduction . . . . . 21
1.1 Research Goal and Questions . . . . . 24
1.2 Scope and Contributions . . . . . 26
1.3 Thesis Structure . . . . . 27

2 Background and Related Work . . . . . 29
2.1 User Support for Continuous Software Evolution . . . . . 29
2.1.1 Software Evolution . . . . . 30
2.1.2 User Support & Feedback for Software Evolution . . . . . 31
2.1.3 The Continuity of Software Evolution . . . . . 33
2.2 Monitoring User's Collective Needs . . . . . 34
2.2.1 The Expectation . . . . . 35
2.2.2 The Perceived Quality . . . . . 36
2.2.3 The Confirmation . . . . . 37
2.2.4 The Contexts . . . . . 38
2.3 User Feedback Analysis Methods . . . . . 39
2.3.1 Rating Analysis . . . . . 40
2.3.2 Textual Review Analysis . . . . . 41
2.4 Research Gap . . . . . 45

3 Research Design . . . . . 47
3.1 Research Structure . . . . . 47
3.2 Case Study as Research Method . . . . . 48
3.3 Data and Analysis Techniques . . . . . 49

3.3.1 Data Gathering . . . . . 49
3.3.2 Analysis Techniques . . . . . 50

4 Results . . . . . 59
4.1 Perceived Quality Analysis . . . . . 59
4.2 Expectation and Needs Analysis . . . . . 62
4.3 Abnormal Update Detection . . . . . 64
4.4 Software Evolution Monitoring . . . . . 66
4.5 Context Analysis . . . . . 70
4.5.1 Situational Context and Ways of Interaction Analysis . . . . . 71
4.5.2 User Type and Preference Analysis . . . . . 72
4.5.3 Software Type Analysis . . . . . 73
4.6 Summary of Publications . . . . . 75

5 Discussion . . . . . 79
5.1 Answering Research Questions . . . . . 79
5.2 Research Contribution . . . . . 81
5.3 Limitation & Future Work . . . . . 84

6 Conclusion . . . . . 87

References . . . . . 89

Publication I . . . . . 109

Publication II . . . . . 119

Publication III . . . . . 131

Publication IV . . . . . 147

Publication V . . . . . 161

Publication VI . . . . . 179

Publication VII . . . . . 191

## List of Figures


2.1 The Expectation-Confirmation Theory [137, 138] . . . . . 34
2.2 Framework of User Feedback Supported Continuous Software Evolution . . . . . 46

3.1 The Research Framework Overview . . . . . 48

4.1 Visualization of the Case Study. . . . . 62
4.2 The Identified Abnormal Days of Whatsapp as an Example . . . . . 65
4.3 Similarity between Update Description and Negative Reviews of Abnormal Days for Whatsapp . . . . . 66
4.4 Hypotheses on Software Evolution Monitoring Mechanism (adapted from Publication II and III) . . . . . 67
4.5 Part of the Similar Topic Chains for Updates of Whatsapp . . . . . 70
4.6 The Detected Communities of Tags . . . . . 75


## List of Tables


3.1 Data Information for each Publication . . . . . 50

4.1 Extracted Topics for Each Review Set . . . . . 63
4.2 Similar Topics Identified among Reviews Before and After the Update 69
4.3 Values of Situational Context Perspectives . . . . . 71
4.4 Detected User Types and Related Features/Preferences . . . . . 73
4.5 Computer Game Types Defined by High-Centrality Tags . . . . . 76

5.1 Summary of Research Contribution . . . . . 82

# ABBREVIATIONS

| | |
|---|---|
| AI | Artificial intelligence |
| API | Application programming interface |
| cf. | compare with |
| e.g. | for example |
| ECT | Expectation-confirmation theory |
| EFA | Exploratory factor analysis |
| et al. | and others |
| etc. | and so on |
| i.e. | that is |
| IS | Information system |
| kNN | K-nearest neighbours |
| LDA | Latent Dirichlet allocation |
| NB | Naïve Bayes |
| NLP | Natural language processing |
| PA | Parallel analysis |
| RE | Requirements engineering |
| RQ | Research question |
| SVM | Support-vector machine |

# ORIGINAL PUBLICATIONS

Publication I — X. Li and Z. Zhang. A User-App Interaction Reference Model for Mobility Requirements Analysis. *International Conference on Software Engineering Advances (ICSEA)*. 2015, 170–177.

Publication II — X. Li, Z. Zhang and K. Stefanidis. Sentiment-aware Analysis of Mobile Apps User Reviews Regarding Particular Updates. *International Conference on Software Engineering Advances (ICSEA)*. 2018, 109.

Publication III — X. Li, Z. Zhang and K. Stefanidis. Mobile App Evolution Analysis Based on User Reviews. *International Conference on Intelligent Software Methodologies, Tools, and Techniques (SoMeT)*. 2018, 773–786.

Publication IV — X. Li, C. Lu, J. Peltonen and Z. Zhang. A statistical analysis of Steam user profiles towards personalized gamification. *International GamiFIN Conference (GamiFIN)*. CEUR-WS. 2019.

Publication V — X. Li, B. Zhang, Z. Zhang and K. Stefanidis. A Sentiment-Statistical Approach for Identifying Problematic Mobile App Updates Based on User Reviews. *Information* 11.3 (2020), 152.

Publication VI — X. Li and B. Zhang. A preliminary network analysis on steam game tags: another way of understanding game genres. *Proceedings of the 23rd International Conference on Academic Mindtrek*. 2020, 65–73.

Publication VII — X. Li, Z. Zhang and K. Stefanidis. A Data-Driven Approach for Video Game Playability Analysis Based on Players' Reviews. *Information* 12.3 (2021), 129.

### *Author's contribution*

The contribution of the dissertation author (i.e., X. Li) within each above mentioned publication is specified as following.

| | |
|---|---|
| Publication I | The author participated in the planning and analysis of the study with the second author who had a significant contribution in structuring the publication. The author also took charge in writing majority of the content under the guidance of the second author. The author is the lead author of the publication. |
| Publication II | The author took a major role in planning the study in terms of conceptualization, data curation, formal analysis, methodology, and validation under the co-guidance of the second and third author. The author also took charge in the writing. The second and third author helped with the editing and reviewing. The author is the lead author of the publication. |
| Publication III | The author took charge in planning the study in terms of conceptualization, data curation, formal analysis, methodology, and validation under the supervision of the second and third author. The author also took charge in the writing with help on the editing and reviewing from the second and third author. The author is the lead author of the publication. |
| Publication IV | The author was in charge of planning the study in terms of conceptualization, data curation, and reasoning under the supervision of the third and fourth author. The second author contributed majorly to the methodology. The author also took charge in majority of the writing with the second author contributing to the rest. The third author had a significant contribution on reviewing and editing. The author is the lead author of the publication. |
| Publication V | The author was in charge of the planning of the study, regarding the conceptualization, data curation, formal analysis, methodology, and validation, as well as the writing. The third and fourth |

author contribute to the reviewing and editing. The author is the lead author of the publication.

Publication VI — The author was in charge of the planning of the study, regarding the conceptualization, data curation, formal analysis, methodology, and validation, as well as the writing. The author is the lead author of the publication.

Publication VII — The author was in charge of the planning of the study, regarding the conceptualization, data curation, formal analysis, methodology, and validation, as well as the writing. The second and third author contribute to the reviewing and editing. The author is the lead author of the publication.

# 1 INTRODUCTION

Entering the new decade, accompanied by the ever-advancing information technology, software products have been further penetrating across people's daily lives. They are more indispensable than ever, as the carriers of services, supporting people's work, easing their communication, encouraging their consumption, and providing them with entertainment. The global software market is gigantic with an influence on most of the global population. Taking mobile application (app) market as an example, in 2018 the global mobile app revenues amounted to over 365.2 billion U.S. dollars when in 2020 the number reached 581.9 with a 59.3% growth [169]. By 2023, mobile apps are projected to generate over 9,935 billion U.S. dollars through paid downloads and in-app advertising [166].

However, despite the rapid growth of the market, software companies are constantly facing challenges in not only the competition from the rivals but also the needs from their customers. As of 2019, over 2 million apps are available for download on the Google Play store, while 1.83 million on the Apple AppStore [166]. However, the retention rate of their users is not optimistic: only 32% of the users returned to an app 11 times and more, with a sharp decrease of 38% compared to the previous year [167]. The situation for users return after 30 days is much worse: the average retention rate drops to 5.7% across all app categories [168]. Therefore, how to deliver to the users the software products that constantly satisfy them and retain their interests is a key issue to all software companies.

Though all users expect to use software products that are perfect, it is inevitable that they have unexpected bugs, faults, features to improve or add, or new environments to adapt to. Thus, software maintenance has long been considered a critical stage of software life cycle [109]. It is also costly and time-consuming [86, 129, 183]. In addition, even decently implemented software systems, especially the ones that mechanize a human or societal activity and are embedded in the real world [93], must be continually adapted; otherwise, they become progressively less satisfactory

[90]. Hence, software evolution being important and inevitable is well acknowledged, the software process models have also evolved towards more evolutionary and agile [7, 13, 28]. However, though software product being evolved to provide continuous quality to users, users' actual needs and expectations are often sidelined or unable to reach the developers [142]. Meanwhile, in academia, various aspects of software evolution have been studied [121, 122]. However, studies on conducting software evolution together with the understanding of what the end users want and the ones on monitoring such activities are still limited.

The support of end users, who provide feedback and requests, is a critical facet of software evolution and shall influence the process [90, 94, 95]. Being the actual consumers, the users provide a view from non technical perspectives different from that of the developers [179]. According to the experiences of software companies, involving users during software evolution and taking their feedback into account shall support the continuous assessment of product acceptance, identify missing features and improve software quality [142]. It is recommended that their requests are continuously considered and used to validate the improved performance, when such a "feedback-validation" loop drives the evolution of software products [51, 179]. However, issues are also found towards effective user involvement, e.g., textual content in the feedback with low quality evokes analysis difficulty; content analysis in large volume is effort-costly; communication between users and developers is disconnected; tool support is limited [142]. Therefore, applying effectively designed data mining and analysis approaches/pipelines is critical towards providing solutions obtaining large volume of user feedback and extracting the latent key information.

Besides the continuous software evolution, assessing and validating the effectiveness of such evolution to sustain users' satisfaction is also critical to the user retention. Therefore, it is important to monitor changes in users' needs constantly during evolution, as well as the confirmation of software quality to such an expectation [19, 29, 137, 176]. According to the expectation-confirmation theory [138, 139], user satisfaction is formed based on the expectations (i.e., needs) and the confirmation of such expectations towards the users' perception of performance, i.e, the software perceived quality [124]. And the users' continuing service (i.e., software) use is determined by the satisfaction with prior use, which is a key to a loyal user base [3]. Thus, to such end, the continuous assessment of the confirmation of users' needs and expectations and their perceived software quality is the key to monitor the software

evolution performance and to detect potential issues.

Hence, an effective process for software evolution shall continuously involve users, track their needs and expectation on the software, assess their satisfaction as well as the changes, and properly address the issues they raise. Though via traditional venues involving a large volume of users for opinions is hard, contemporary software distribution platforms enable users to provide ratings and reviews reflecting their needs, expectations, praise and needs [78]. Many studies conduct empirical studies on the ratings of software products towards understanding the nature of such user behavior and that of the rating mechanism itself [24, 36, 41, 59, 80, 143, 155, 160, 200]. Compared to rating feedback, textual reviews from the software users can provide much more information, including feature requests, functional errors, hidden costs, uninteresting content, and so on [78]. Many studies also propose using a large volume of user reviews to analyze the determinants of review helpfulness [26, 85, 126, 146], automatically assess the review quality with machine learning and topic modeling methods [47, 72, 81, 111, 125, 159, 186], extract features [20, 57, 66, 148] and design and implement tools supporting such automated analysis [25, 37, 45]. To a large extent, these studies greatly address the challenges for user involvement in supporting software evolution.

Furthermore, when using large volume of user feedback data, it is also critical to understand and consider the differences amongst the users and the situations where they use the software system. Therefore, besides the understanding of users' needs, it is also important for the developers to take into account the contextual information via systematic analysis so that the planning for maintenance activities shall be enhanced. Due to the adaptivity embraced by software evolution, software system providers should understand the context information of the system being used and the information of the users who use the systems [82]. Any such information that can be used to characterize the situation of the person, place, or object, is considered relevant and important to the interaction between a user and the system, including the user and applications themselves [1]. Specifically, the context information, e.g., the surroundings of the users, physical or social [102], the traits of the clustered user groups [99] and the types of software products [98, 100] is an important supporting indicator to facilitate relevant decision making practice in the software maintenance.

## 1.1 Research Goal and Questions

Though the end users' needs and satisfaction is a critical measure and reflection of the quality and success of software products, the studies on evaluating and monitoring this measure during software evolution are limited. For software evolution, user involvement and support with feedback is critical; however, the difficulty in involving large volume of users and effectively extracting their opinions from low-quality feedback remain. Such a research gap can thus be mended by introducing data-driven and NLP methods. Meanwhile, though many studies work on the NLP-based data-driven methods on analyzing large volume of user reviews, monitoring the changes of users' expectation and satisfaction within software evolution is not covered. Furthermore, though context is considered important influencing users' perceived quality in use of the interactive software systems, data-driven methods for context analysis are still limited.

Therefore, towards mending the research gaps mentioned above, the main goal of this thesis is to investigate the application of data mining techniques towards facilitating the continuous monitoring of software evolution status in terms of the users' changing needs and perceived software quality. The proposed series of data-driven approaches aim to ease the effort of the developers in continuously monitoring the status of updates fulfilling the needs of their end users. Towards achieving such goals, the following critical aspects are taken into account within the research.

1. The quantification and evaluation of the collective perceived quality of a particular software product.
2. The identification of the merits and defects of the software regarding a particular update.
3. The identification of the problematic updates that the users do not like.
4. The tracking on the changes of the above aspects.
5. The additional information that can be extracted towards facilitating the activities mentioned above.

Therefore, this thesis, with all the aspects of objectives mentioned above, shall contribute to the answering the following research questions (RQ) respectively.

- **RQ1. How to analyze users' collective expectation and perceived quality in use with data-driven approaches by exploiting sentiment and topics?**

  Firstly, it is critical to explore the methods to evaluate users' perceived quality for a particular software product and extract their collective expectations and needs. Provided a large volume of user review text data is obtained, the first key task is to extract users' opinions and detect the strength of their opinions on the key aspects of the software quality. An effective way of extracting information from large volume of text data is via topic modeling when this research shall explore and verify how topic modeling methods can be used to facilitate the analysis of users' needs. Meanwhile, the sentiment of the users' feedback text can reflect the software's perceived quality. Thus, herein, the research shall also explore how sentiment analysis can be applied towards the evaluation of software perceived quality.

- **RQ2. How to monitor user satisfaction over software updates during software evolution using reviews' topics and sentiments?**

  Secondly, to facilitate the continuous evolution of software products, the users' satisfaction shall be monitored closely, especially when new versions are released. Thus, with the method proposed to answer RQ1, this research shall further explore the ways of measuring the changes of users' expectations and their perceived quality. An ideal way of achieving such result is finding a way to detect the changes in users' opinions, e.g., when the issues complained by the users previously get solved afterwards. Provided the reviews on different software versions are obtained, the research aims to provide a method to detect which are the users' needs that are changed between different versions.

- **RQ3. How to analyze users' profiles, software types and situational contexts as contexts of use that supports the analysis of user satisfaction?**

  It is also worth noting that the different contexts of use shall influence the perceived quality and user expectation and needs. Therefore, this research aims to address the question on how to use data-driven methods to analyze important context information. Regarding users' profiles, the research shall answer how to classify different types of users based on their behaviors with the software. On software types, it shall also address the issue of how to classify different software products based on their features. Furthermore, regarding the situa-

tional contexts, the question is how to detect the software's "uncomfortable designs" when being used in different contexts.

## 1.2 Scope and Contributions

Regarding the scope of the research, the effort of this thesis contributes to multiple interconnected research areas. Firstly, the thesis proposes a practical data-driven approach to effectively monitor software evolution in practice. It provides insights and potential solutions on one of the key issues of software maintenance and evolution, i.e., continuously provide user-satisfied software updates by knowing in time how good the software is, when the problems occur, what they are, and when they are solved. Ideally, for any individual software product in its evolution phases with sufficient volume of user reviews and the timeline of its updates, the approach shall enable developers to evaluate and visualize the perceived quality of the software in terms of a selected quality framework. Furthermore, it also helps them be aware of the changes in users' complaints and satisfaction on the software in general and specific updates, especially the ones that evoke severe negativity. Secondly, this thesis also provides extended insights on how the context of use information regarding either the software products or users can be analyzed facilitating such a solution, where the updates shall solve users' needs more precisely.

Moreover, the contribution of the thesis can also be seen as an interdisciplinary study towards data mining application in the domain of game studies [118]. It certainly sheds light on the way of understanding the players' perceived game quality (i.e., playability) [35]. Nonetheless, such a promising extensible sub-scope is not the focus of this thesis but an important part of the future work. To be noted, this research focuses on the mass-market software (e.g., mobile apps or computer games on online distribution platforms) instead of customer-oriented software systems. The reason for such selection is the accessibility of the relevant data. Nonetheless, the proposed approach is considered applicable to customer-oriented software systems given adequate review data obtained.

## 1.3 Thesis Structure

The remainder of this thesis is organized as follows. Chapter 2 presents the background of the relevant research domains, including, software evolution, user expectation, and data-driven feedback analysis. In addition to the briefly introduced background of each field, the further respective introduction of the related works is also presented, especially regarding the studies with similar contributions. Chapter 3 describes the design of the research, as well as the strategies and approaches applied in this work. Chapter 4 presents the results of this research, including the concluded answers to the above-raised research questions and the summary of each publication. Chapter 5 revisits the results and contributions of the thesis and further discusses the potential future works that could conduct further impact. Chapter 6 concludes the thesis. Additionally, the selected original research publications are reprinted and attached in their original format at the end of the summary section.

# 2 BACKGROUND AND RELATED WORK

In this chapter, Section 2.1 firstly revisits and briefly introduces the background of the software evolution research domain in general. Specially, this section explains the fact that software evolution being continuous and requiring the support of end-users to enhance its effectiveness and presents the related works in this area. Thereafter, due to the importance of user satisfaction to the success of software evolution, Section 2.2 introduces the theoretical foundation on the determinants of user satisfaction. Therein, the importance of understanding such determinants, e.g., users' expectations and the perceived quality of software products, is emphasized. Additionally, the section also proposes the collective analysis on such determinants, reflecting the statistical representativeness, based on a large volume of user feedback and introduces the related works. Furthermore, Section 2.3 focuses on acknowledging the related methods proposed to support such collective feedback analysis using statistical and data mining techniques. Lastly, Section 2.4 synthesizes the presented related works and compares the difference between the contribution of this thesis and theirs.

## 2.1 User Support for Continuous Software Evolution

Software maintenance and evolution has long been considered an indispensable part of software life cycle and critical practice. Due to the inevitable negative effect of software aging, effective and on-time support for software evolution is important [122]. Emphasized by Lehman regarding the nature of software evolution, users' support, specifically their feedback information and requests, is a critical facet and shall influence the process [90]. On the other hand, the benefits of user involvement in software engineering process have also been acknowledged widely [87]. However, regarding the user involvement practice in software evolution, despite its importance for developers on software quality improvement and missing features identification,

the consolidation, structuring, analysis, and tracking of user feedback is still effort-consuming [142].

### 2.1.1 Software Evolution

Software maintenance is defined as the modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment [67]. The main categories of maintenance include corrective, adaptive, perfective and preventive, which are still commonly adopted [71, 110]. It has been commonly acknowledged that software maintenance, especially as a fixed final step of the software life cycle [191], is both inevitable and costly [86]. Comparatively, though were used interchangeably with the term "maintenance", software evolution is seen as *an alternative to describe various phenomena associated with modifying existing software systems* [50]. The difference between software maintenance and software evolution is threefold: 1) Maintenance suggests preservation and fixing, whereas evolution suggests new designs evolving from old ones; 2) Maintenance is a set of activities conducted *on* the system, whereas evolution concerns whatever happens *to* a system over time; 3) maintenance is focusing on engineering goal, whereas evolution is concerning a broader and more scientific nature [50].

Specially focusing on software evolution, Lehman and collaborators are the pioneers studying its fundamental difference from software maintenance. Lehman formulated a set of observations named *Laws of Evolution*, focusing on E-type programs, i.e., the ones that mechanize a human or societal activity and are embedded in the real world [90, 91, 92, 93, 94, 95]. To simply and informally put, an E-type program are software that implements an application or addresses a problem in the real world, e.g., instant messenger mobile applications for communication, video games for interactive entertainment, and so on. Regarding E-type programs, according to Lehman, the processes for E-type programming constitute *multi-loop, multi-level feedback systems* [90]. In addition, Lehman emphasizes users' support, specifically their feedback information and requests, is a critical facet and shall influence the process [90]. On the other hand, such a multi-loop feedback mechanism is also reflected in the staged model for software lifecycle, where the evolution stage is driven by multi-rounds of changes [7]. Therein, when outlining such a model, spe-

cially on software evolution, Bennett and Rajlich emphasize the knowledge of user requirements is a crucial prerequisite and during the evolution applications adapted to such ever-changing user requirements is the key [7].

### 2.1.2 User Support & Feedback for Software Evolution

Alongside the emphasized importance of software evolution, it shall be noted that the end-users of a software product are the actual consumers whose interests are in its functionality from a non-technical perspective [179]. It is recommended that their requests towards changes are taken into account by the developers who also need to know whether the corresponding implementation has addressed users' needs and has improved the performance [179]. Therefore, as time passes, when the software product inevitably evolves, the end-users and the feedback information they provide become part of the feedback loop driving such continuous evolution [51]. Such a notion of user involvement, even before its connection to software evolution, has also been long studied towards maximizing the usefulness and usability of software products by understanding users' needs and expectations [87]. It has become increasingly important to involve end-users within the process of software development and evolution. One critical reason is that the distance between developers and end-users is getting larger due to the shift of the majority of users to the non-technicians and the changes of distribution channels to online software platforms [55, 56]. Therefore, the developers shall pay extra attention to the changing requests from the users, especially during the post-deployment phase [84, 115].

Although the awareness regarding the importance of users' feedback in software evolution is raised, early studies focused on the user involvement in early stages of the software lifecycle, the degree and approach of such involvement, and the according effects on software acceptance [87, 185]. Specifically, regarding the factors that hinder software evolution decisions, Ko et al.'s study shows 1) that developers prefer to cope with the feedback from the majority of the users, and 2) the disagreement in users' feedback also hinder any feedback from being taken into account [84]. On the other hand, Heiskari and Lehtola investigate the state of user involvement practice back then and find that 1) user feedback and relevant information was difficult to access with no explicit process of understanding users, 2) it is hard to determine the average end-user opinion, and 3) the integration of user knowledge into develop-

ment and evolution process is needed [61]. Zimmermann et al.'s study on open source software bug reports concurs that it is hard for developers to understand the users' needs with missing and mismatched information and indicates that well-known users' opinions, even less important ones, are more likely to be taken into account [199].

Pagano and Bruegge conduct an empirical case study on five software companies focusing on exploring the practice of user involvement specifically during the software evolution stage [142]. Part of their findings confirmed the conclusion of the previously mentioned studies. In addition, their study investigates also the multiple aspects of the user involvement practices in software companies, including infrastructure, frequency, communication, motivation, analysis methods, problems, tool support, consolidation, and assessment. Besides the findings confirming the previous studies, Pagano and Bruegge's study presents the following indications concerning the understanding of user feedback: 1) users tend to choose more public channels to publish critical feedback; 2) users frequently provide feedback which does not always reach developers; 3) user feedback supports continuous assessment of product acceptance; 4) user feedback can improve software quality; 5) user feedback helps to identify missing features; 6) user feedback conveys trust in the form of positive user ratings and user experience; 7) developers prioritize feedback-oriented tasks based on their occurrence frequency.

In addition, regarding the existing problems of user involvement practice, Pagano and Bruegge find that 1) natural language content, low quality and contradiction in user feedback evokes analysis difficulty; 2) content analysis, impact, and priority estimation are effort-costly; 3) user-developer communication channel is disconnected; 4) tool support to structure, analyze, and track user feedback is needed, particularly when feedback volume is high [142]. However, these uncovered problems can be properly solved by applying data mining, machine learning, and natural language processing (NLP) techniques and the wide adoption of online software distribution platforms. For example, Chen et al. use Expectation Maximization for Naïve Bayes (EMNB) [133] to extract informative feedback from the raw textual content, which eases the feedback analysis [25]. Villarroel et al. use the DBSCAN clustering algorithm [40] to prioritize the user reviews facilitating the planning of subsequent mobile app release [184]. In addition, online software distribution platforms, such as Apple App Store and Google Play for mobile apps, Steam for video games, and Mi-

crosoft Store for Windows Desktop software, are providing communication channels connecting developers and users. For the large volume of user feedback data from such platforms, many tools are proposed in academia facilitating the understanding of user feedback [46].

### 2.1.3 The Continuity of Software Evolution

On the other hand, besides the requirements of user support and feedback, an E-type program must be continually adapted, otherwise it becomes progressively less satisfactory with its functional content must be continually increased to maintain user satisfaction over its lifetime [90, 91, 92]. Hence, the continuity of software evolution is also a critical aspect. Within a larger scope, the trend towards continuous software engineering has been widely reflected within the need for flexibility and rapid adaptation in software development environment [43]. Such vast interest across the industry has been raised since the spreading recognition of the importance of increased frequency of key activities, from the notion of '*release early, release often*', to the concept of DevOps, to the adoption of agile methods, to the thinking of lean [33, 43, 79, 96, 147]. The overall continuous activities in the entire software life cycle positioned within a holistic view is proposed as the notion of *Continuous**, which encompasses the continuous activities in business strategy and planning, development and operations [43]. Therein, continuous software evolution is anchored as one of the key activities in the development phase.

Specially on software evolution, according to Lehman's laws, for most of the contemporary software products (i.e., E-type programs), the nature of being continuously adapted and improved is required to sustain satisfactory quality [90]. Accompanied with such nearly inevitable continuous evolution, the complexity of software is increasing with functional content increasing as well. Though the concept "discontinuity" of software evolution is also brought up, meaning the required changes to evolve are beyond the level of tolerance to change so that the architecture requires re-engineering, the notion of continuously required improvement in software evolution remains [4]. Specially, the evolution of mobile apps is fast due to the rapid technology evolution and the online distribution service [4]. Furthermore, based on the difference in update time span and pace of introducing new features, the evolution of mobile apps can be seen as emergency-driven, feature-driven or pace-driven

[103].

Overall, the support and feedback from end-users are critical towards the long-lasting success of software products via continuous software evolution. Though it is difficult for developers to efficiently and effectively involve a large number of end-users and extracting their requirements [143], data mining and NLP techniques applied on user feedback data from online software distribution platforms shall contribute to the solution. Meanwhile, it is also important to sustain the continuity of the evolution process in order to continuously satisfy the users [90]. Thus, when extracting user requirements from feedback, the developers should also be aware of whether and to what extent these needs are addressed through updates.

## 2.2 Monitoring User's Collective Needs

Due to the rapidly growing software market and its competitiveness, how to deliver to the users the software products that constantly satisfy them and retain their interests is a key issue to all software companies. Therefore, a general investigation of user satisfaction is necessary. The expectation-confirmation theory (ECT) is widely cited towards understanding the satisfaction of customers in the marketing literature [137, 138]. ECT indicates that the customers repurchasing particular products or continuing service use is determined by their satisfaction with prior use of the product or service, which is also a key to retain such a loyal base of long-term customers [3, 138].

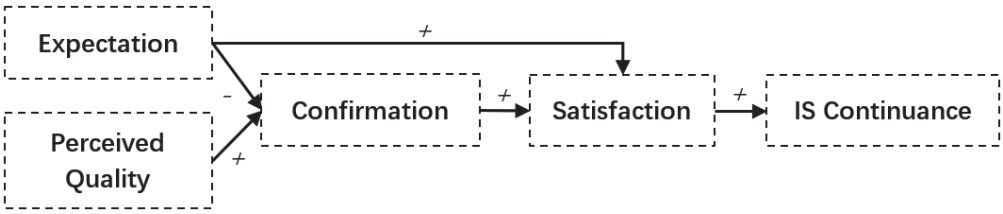


**Figure 2.1** The Expectation-Confirmation Theory [137, 138]

The process towards customers' repurchase intention can be described as follows (shown in Figure 2.1): Firstly the customers form the initial *expectation* towards a particular product or service before purchasing it. After using it, the customers form the *perception of the quality*. When such perceived quality is assessed according to their expectation, the extent to which the expectation is confirmed is deter-

mined. The *satisfaction* is formed based on the expectation and the corresponding *confirmation*, which leads to the potential repurchase intention or discontinuation of subsequent use. Regarding software products, the identical conclusion is drawn by Bhattacherjee [10] who also emphasizes that due to the fact that post-acceptance satisfaction is grounded in users' first-hand experience with the software, ignoring it can have disastrous consequence for user continuance.

### 2.2.1 The Expectation

Generally, expectations are consumer-defined probabilities of the occurrence of positive and negative events if the consumer engages in some behavior [139]. It is the wished calculation of probability of performance or the expected performance levels by the customers [174]. In information system (IS) studies, Szajna and Scammell define the expectation of IS users as *"a set of beliefs held by the targeted users of an information system associated with the eventual performance of the IS and with their performance using the system"* [176]. They also emphasize that the realism of user expectations is potentially a factor in the success or failure of IS. Therefore, user involvement in the design of an information system brings about realistic expectations of system capabilities [48]. And it is also an important job of software project managers to make sure such expectations are realistic and consistent with the promised software deliverable [5, 176]. The unrealistic expectation is somehow one of the critical risks of software project management when it can sabotage both the success of products and projects [5].

Regarding the difference and connection between the terms, expectation, and requirement, it is common to see them as identical, as *in a broad sense, every requirement is also an expectation: the customer expects that the requirement will be met* [187]. However, the main difference between them is that *the former are objectively stated and clearly defined, while the latter are subjective, and loosely understood* [187]. Hence, in terms of RE, user requirements is a linking concept towards "users' needs" adopted in both IS study and software engineering [112, 116], especially when end-user is a critical stakeholder [134]. The traditional methods of identifying user requirements include surveys, interviews, focus groups, scenarios and use cases, workshops, and evaluating an existing or competitor system [116]. These methods are also adopted in requirements elicitation of user-centered requirements engineering practice [171,

197]. To be noted, the outcome of such elicitation is rather the raw expectation of the users, as requirements specification is needed to transform such subjective and ambiguous expectation into objectively described and documented requirements [116].

Thus, towards the assuring of user expectation/requirements being understood, the involvement of end-users in RE process is critical for the quality of the requirements, and furthermore, the success of the software [88]. Meanwhile, it is common that such requirements elicitation traditionally has to satisfy the needs of the majority of users [172]. However, the traditional requirements elicitation methods fall short in involving a sufficient amount of end-users which is representative of the majority. Hence, approaches to collect spontaneous end-user feedback are proposed [162, 163]. Furthermore, adopting the idea of *Wisdom of the Crowds* [170], crowd-based requirements engineering (CrowdRE) is proposed as *a semi automated RE approach for obtaining and analyzing any kind of "user feedback" from a "crowd", with the goal of deriving validated user requirements* [53, 54]. Specially emphasized therein, Groen et al. indicate that CrowdRE should cater to the diversity of users despite their backgrounds and expectations [54]. Hence, eliciting and analyzing a large volume of end-user (i.e., crowd) feedback from social media and app stores is the key for Groen et al.'s approach; however, they fall short in providing specified techniques supporting such an approach [54].

### 2.2.2 The Perceived Quality

The perceived quality is a critical factor determining the confirmation of IS users' expectations and furthermore, their satisfaction [10, 138]. It indicates the degree to which a person believes that using the system shall enhance his/her performance and reflects how systems (should) behave in order to meet users' constraints and needs [38]. Knowing and being able to predict such perceived quality may allow an organization to adjust deployment to meet the quality expectations of its customers [124]. Many studies have explored the factors determining the users' perceived quality of interactive software products and the potential dimensions of the concept [38, 124]. On the other hand, considering the broader concept of perceived quality in the domain of service marketing [52], such perception of software quality measured in terms of the result of using the software rather than properties of the software itself is referred to as the software product's *quality in use* [8, 68]. Specially, compared to

the internal and external quality of software, its quality in use is measured by the extent to which the software meets the needs of the user in the working environment [8].

Regarding the measurement of product quality in general, several models are proposed. For example, the early *Factors-Criteria-Metrics* model provides a fundamental goal-oriented methodology for measuring software quality with a set of questions used to measure each individual criterion [119]. Similarly, models using low-level metrics instead of questions towards objectively measuring attributes to higher-level characteristics are also proposed [14]. The major defect of such approaches lies in the difficulty to obtain all metrics from "bottom-up" towards a global measure of quality which can be achieved via surveys [189]. In Xenos and Christodoulakis' study on the perceived software quality measurement, they propose a survey-based perceived quality measuring method that focuses on the "satisfaction of customer requirements" as software perceived quality [189]. However, the disadvantages of the survey method include the cost in deploying the techniques, error rates, subjectivity of the answers, and human factors in surveys and measurements [189]. On the other hand, heuristic evaluation, as another external measurement method, is also widely adopted; however, such a method relies on the experiences of experts heavily and can cause bias according to the changing mindsets of evaluators [132]. Machine learning methods are also applied to evaluate software usability, e.g., for eLearning systems in [140]; however, the method is neither verified extendable to other quality attributes nor to other software types.

### 2.2.3 The Confirmation

According to the ECT, user satisfaction is determined by the extent to which his/her expectation of the perceived quality of the software product is confirmed [10, 138]. Such activity of confirmation is important, because the mismatch between customers/users' expectations and the features provided by the software product is one of the main reasons for the failure of many software projects [154]. In RE domain, requirements traceability is the activity aiming to confirm and ensure the continued alignment between user requirements and system evolution, where the satisfaction of requirements is one of the major focuses [152]. Specifically, a goal-oriented RE modeling approach is used towards the specification on what to be satisfied and con-

firmed [31]. To such an end, Letier and van Lamsweerde's early study introduces a semi-formal application-specific approach specifying the partial goal satisfaction, where propagation rules and refinement equations are used to calculate the satisfaction degree [97]. Furthermore, Holbrook et al. apply TF-IDF similarity score and textual similarity towards the automation of requirements satisfaction assessment [62]. Port et al.'s study also uses text mining as support towards the confirmation of software requirements in terms of traceability assurance [150]. However, these approaches focus on the assurance and confirmation of requirements in terms of high-level business goals in the form of specified and well-formatted requirements. Therein, limited studies have explored the confirmation of end users' satisfaction towards the evaluated end-users' perceived quality.

### 2.2.4 The Contexts

As software evolution embraces adaptivity when system providers need to understand the context where the systems are used and the context of users for the adaptation as well [82]. An operational definition of context is that it *is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves* [1]. Many studies also propose the categorization of context types which facilitates the potential context-aware designs when *location*, *identity*, *activity* and *time*, *social connections*, etc. are the ones mentioned by the majority [1, 157, 161]. Regarding the context of use for usability, the main attributes of which include: 1) the *user*: a person who interacts directly or indirectly with the system, 2) the *task*: a piece of work that the user carries out by interacting with the system, and 3) the *environment* in which the system is used, i.e., the external factors that affect the use of the system [2, 9, 69]. On the other hand, it is inevitable that the users' perceived quality in use is measured by the users within its contexts of use [68]. Thus, when assessing it as well as the confirmation of it to the users' expectation, care shall be taken when such assessment is generalized to another context with different types of users, tasks or environments [8]. Specifically, in order to achieve such quality in use by incorporating human-centered design activities throughout the life cycle of interactive systems, understanding and specifying the context of use is one of the key tasks [8, 70].

Therein, the situational context refers to the extrinsic properties of the user and the software that impact the initiation of interaction between the user and the software, mostly mobile applications [102]. Specially regarding the use of mobile applications in general, situational context is an important factor that influences the adoption of mobile applications and the according users' behaviors [108, 181, 190]. Many previous studies have explored the multiple aspects of the situational contexts of mobile application use, including the definition of the interaction tasks, the physical environment, the user and social environment, and the types of the application in use [30, 74]. Though the concept of situational context is studied, the studies on taking into account such a concept within the process of software evolution are limited. Specially, user profiles and their preferences are also important context information supporting the potential personalized design of software products [173]. With such context information, user satisfaction can be enhanced via a recommender system designed for personalized services [107], as it can be seen as either new features or enhancement within the evolution stage. Furthermore, the types of the application are also studied as one of the key contextual factors [30, 74]. However, the ways of analyzing the existing software types based on the provided features using data-driven approaches are not adequately explored.

As mentioned previously, the ECT shows the simplified causal relation between end users' expectations and perceived quality of software and their satisfaction. It can be seen as the main goal of software evolution to maintain the satisfaction of the majority of users by confirming their changing expectations during the evolution lifecycle. Many studies in the last three decades have investigated the critical elements of ECT, i.e., user expectations, perceived quality, and confirmation, respectively in the domain of software engineering. However, studies on the continuous monitoring of collective user satisfaction in order to improve the software evolution practice with ECT taken into account.

## 2.3 User Feedback Analysis Methods

In order to enhance user involvement towards effective software evolution, it is important to obtain the users' feedback on their acceptance of the systems. The traditional feedback channels include email, forum, and social media when such data

is hard to be extracted and synthesized without pre-implemented API. Understanding the importance of user feedback, software companies inject the feedback feature in their online distribution platform, e.g., Google Play[1], Apple AppStore[2], Steam[3], PlayStation Store[4], Microsoft Store[5], etc. Therein, the users can submit their feedback regarding the target software in form of ratings (either numeric ratings or recommend/not recommend) and textual reviews [143]. The analysis of such feedback information is critical to facilitate the evolution of software products and services [17, 83]. Therein, many studies have provided approaches to analyze such information in order to support the sustainability and improvement of software products.

### 2.3.1 Rating Analysis

In general, the ratings of software products and their popularity are certainly correlated, so that the developers must carefully take into account such evaluation results [42]. Therein, empirical studies on the rating feedback that reflects the nature of such user behaviors or that of the rating mechanism is common in academia.

Pagano and Maalej's empirical study on 1,126,453 reviews from 1,100 mobile applications of Apple AppStore shows that the average rating is very positive when the apps' ranking and the ratings are not independent of each other [143]. Regarding the quality of the feedback, the study also finds that when the helpfulness of feedback is assessed by the user community, such assessment tends to be either very positive or very negative. Relevant results are also provided by Chen and Liu's study via observation, indicating that the top-ranked paid applications are not necessarily closely correlated with the ratings given by the users [24]. Harman et al.'s study on the Blackberry Store also provides results on the correlations between software ratings and other key factors: the ratings of apps are not correlated with the prices but are strongly correlated with the numbers of downloads [59].

Ruiz et al. analyze the ratings of 242,089 app versions of 131,649 selected mobile apps towards examining the rating system mechanism of mobile app stores [155]. Based on the statistical analysis on version ratings, they suggest that developers have

[1]https://play.google.com/store/apps
[2]https://www.apple.com/app-store/
[3]https://store.steampowered.com/
[4]https://store.playstation.com/
[5]https://www.microsoft.com/en-us/store/apps/windows

no incentive to maintain or improve app ratings and gain limited benefits when version ratings are not displayed. They also suggest that more advanced ways to generate ratings, e.g., exponentially decaying the older ratings and thus emphasizing the recent version ratings, should be explored.

Sarro et al.'s study on 11,537 mobile apps from Blackberry and Samsung AppStore with 1,876 features shows that apps' rating is predictable from their claimed features [160]. In addition, the study also shows that for all app categories, the ratings are predictable for 98% of the cases regardless of the app sizes. They also validate that the performance of such prediction is not influenced by the configuration of the case-based prediction system when the best prediction can be achieved with only a limited number of apps.

Finkelstein et al.'s study also confirms the correlation between mobile apps' ratings and their number of downloads, but still little evidence on that between price and ratings [41]. The study also shows that free mobile apps are generally rated higher and more popular than non-free apps.

Furthermore, many other studies also provides insights on the connection between the users' ratings and the other relevant attributes of software products, as well as reflecting other user behaviors. For example, Di Sorbo et al.'s study suggests that users experiencing bugs tend to provide lower ratings when no such impact is observed for users asking for new features or enhancement [36]. Zolkepli et al.'s research suggests that users give higher ratings to mobile apps that are trending and would prefer to pay for those with higher ratings [200]. Kim et al.'s study shows that a mobile app's rating is one of the key determinants in a user's purchase decision [80]. However, despite the studies showing the importance of user ratings and suggesting such information being taken into account, due to limited information provided therein, studies that use only rating feedback to support software engineering practice are limited.

### 2.3.2 Textual Review Analysis

Compared to rating feedback, the textual reviews from the software users can provide much more information, including feature requests, functional errors, hidden cost, uninteresting content, and so on [78]. This information is thus critically meaningful to the developers or product owners' understanding of the collective expecta-

tion of the users, provided such information is elicited from a large volume of review data.

Based on Pagano and Bruegge's research, the quality of textual user feedback is an important factor challenging the involvement of users in software evolution [142]. Regarding the quality of user reviews, an early study by Cheung et al. confirms that information usefulness has a significant impact on consumer decision of adopting such information [26]. Their findings also admit that the accuracy of such information quality is hard to evaluate by the users. From a different perspective, Mudambi and Schuff's study shows that review depth has a positive impact on the perceived helpfulness of the reviews when the extremity of reviews helps only for search goods rather than for experience goods, e.g., majority of software products [126]. Regarding the determinants of review helpfulness, Pan and Zhang find review length and positive reviews are directly correlated to review helpfulness [146] when Korfiatis et al.'s findings convey similar conclusions and add review readability into such determinant list [85].

Moreover, towards the assessment of user review quality, an early study of Kim et al. proposes an automated assessment approach to use support vector machine (SVM) to predict the helpfulness of unlabelled reviews using both numeric ratings and semantic features of review texts [81]. Comparatively, Ghose and Ipeirotis' study applies a Random Forest-based classifier to automatically identify reviews expected to be helpful based on reviewers' relatedness and subjectivity as well as review readability [47]. For a similar purpose, Moghaddam et al. use a probabilistic graphical model based on Matrix Factorization and Tensor Factorization, focusing on the latent features [125]. On the other hand, identifying spam reviews is another way of enhancing the quality of the user review. For example, Jindal and Liu build a logistic regression model using both the textual features and rating features of reviews to identify different types of spam reviews, including untruthful reviews, brand-only reviews, and non-reviews [72]. Comparatively, other models and approaches are also used for the same purpose, such as linear regression model [111], heterogeneous graph model [186], as well as semantic similarity method and LDA model [159].

Within user reviews together with the associated sentiments [57], Park et al.'s study uses adapted LDA topic modeling method to analyze both user reviews and mobile app descriptions in order to elicit important features of apps by bridging the vocabulary gap between developers and users [148]. On the other hand, statistical

analysis methods are also used to extract features from user reviews. For example, Pagano and Maalej's study uses the distribution of the number of topics per feedback interpreting the main themes from user reviews [143]. In Fu et al.' study, descriptive statistical analysis is used to facilitate the interpretation of market trends as well as the extracted aspects of mobile app quality [45]. The other approaches applied towards feature extraction of textual user reviews include SVM used by Oh et al. extracting functional and non-functional user requests [135], Naïve Bayes and Decision Tree classification approaches used by McIlroy et al. extracting and labeling different user issues [120], even manual tagging used by Khalid and colleagues extracting and summarizing user complaints [77, 78], and so on.

Tools to support automated analysis of large user review volume are also proposed. Fu et al.'s study proposes an analytic tool, WisCom, which provides insights based on mobile app user textual reviews [45]. The tool enables discovering inconsistencies in reviews, identifying reasons why users like or dislike a given app, and provide an interactive view of the evolution of reviews over time, and identifying users' major concerns and app preferences. The results are demonstrated via the over 13 million user reviews of 171,493 Android apps in the Google Play Store. Chen et al. propose the AR-miner framework that facilitates extraction of informative user reviews by filtering the irrelevant, review grouping by topics, review prioritization by review ranking scheme, and presentation of the informative review groups via visualization [25]. Di Sorbo et al.'s study introduces the Summarizer of User Review Feedback approach to capture the users' needs and recommend the changes needed by the users [37]. Such analysis of user reviews provides the developers with insights on understanding the maintenance tasks and identifying which parts to change.

To sum up, for nearly all the contemporary mass-market software products, evolution is inevitable and critical for their long-term acceptance and profitability. The maintenance effort within the evolution phase shall be continuous in order to meet the changing needs of the majority of end-users. However, despite being emphasized by Lehman that the support of users' feedback is important, the studies to support such activities are still limited, where this thesis aims to contribute. Comparatively, Pagano and Bruegge's study raises the awareness of such a research gap as well as the challenges within but does not provide solutions [142]. Subsequently towards such solutions, Pagano and Maalej also find utilizing the convenient online distribu-

tion platforms is useful towards user involvement but did not fit such a study in the software evolution domain [143]. Following this thread, many studies that propose approaches of analyzing user feedback and extracting information also fall short in mapping their contribution to software evolution [25, 45, 57]. On the other hand, research of software evolution focuses more on improving the software quality from the module level and source code changes [64, 131, 198] and the evolution practices for specific software types or particular services [18, 60, 158, 194] with very limited attention paid to the investigation of users' opinions. Regarding the effectiveness of software evolution, the satisfaction of end-users is a critical metric. According to ECT, the customers' continuous use of the software is determined by their satisfaction, which is determined by the extent to which their expectation of it is confirmed by their perceived performance [10, 137, 138]. However, the theory, as well as the key conceptual components, is seldom adopted in software engineering, especially regarding software evolution, though many studies in the information system research domain explore the known determinants and other relevant variables [39, 114, 165, 178].

Therefore, specification of the users' expectations and needs, as well as that of their perceived quality of the software product is critical. Regarding the measure of software perceived quality, the survey and statistical approach, e.g., the one proposed by Xenos and Christodoulakis [189], is certainly useful but yields to the limitation, including the cost in deployment, error rates, subjectivity, and human factors. On the other hand, regarding the extraction of users' expectations and needs, it is impractical to apply traditional requirements elicitation techniques, such as interviews or observation studies [182]. Comparatively, opinion mining from user feedback towards both perceived quality evaluation and user expectation extraction is more efficient and accurate. Palomba et al. propose CRISTAL approach tracing informative crowd reviews and meanwhile monitoring the extent to which developers accommodate the crowd requests [144, 145]. The CRISTAL approach contributes to the mapping and tracing of reviews and the code changes together with the measure of users' satisfaction towards such changes. Instead of the mapping to source code, this research explores the use of sentiment analysis towards perceived quality measurement and abnormal update detection that also supports the software evolution practice.

## 2.4 Research Gap

The research gap between this thesis and the existing work can be seen from the following three perspectives.

1. *Difficulties in involving end users and extracting their feedback in software evolution.* It has long been emphasized that user support and feedback is critical to the continuously evolving software system from a non-technical perspective, when the studies towards finding effective ways of utilizing such information are still limited. Though the traditional ways of involving end users in software evolution are studied, difficulties in extracting representative opinions remains in inviting large number of users for surveys and also in analyzing feedback content with natural language and in low quality. Hence, the data-driven approach proposed by this research aims to mend such gap.
2. *Lack of methods in continuous monitoring evolution status via the changes in users' collective needs.* Many existing works have proposed the NLP-based data-driven methods on analyzing large volume of user reviews. However, those studies have not sufficiently addressed the issues in monitoring the status of software evaluation in terms of the changes in users' perceived quality and satisfaction. Guided by the commonly acknowledged ECT as theoretical foundation, the research shall provide a way of checking the confirmation of users' collective expectations and tracking the changes of their satisfaction through the evolution.
3. *Lack of methods in analyzing the various context information.* Users' context information is important influencing their perceived quality in use of the interactive software system. Though many studies have investigated the definition and classification of such context information, the analysis methods in data-driven fashion are limited. This research also aims to mend the gap towards analyze the fundamental but important context information using data-driven and statistic methods.

Hence, this thesis aims to contribute to mending the research gaps mentioned above focusing on effective user involvement and feedback analysis, continuous monitoring evolution status based on collective user expectation confirmation, and data-driven context analysis. Illustrated in Figure 2.2, the main focus of the research is to

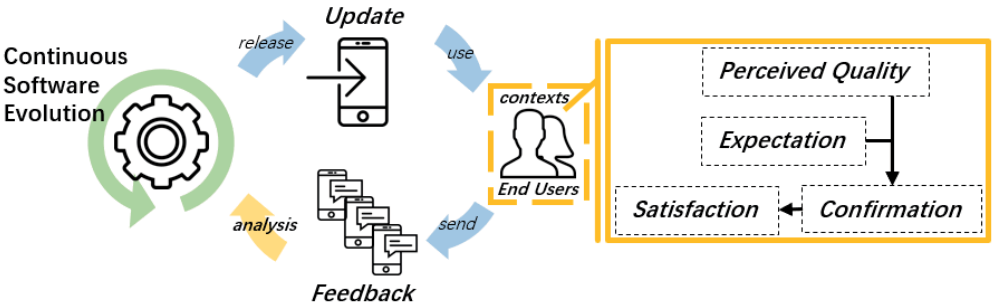


**Figure 2.2** Framework of User Feedback Supported Continuous Software Evolution

explore the ways of evaluating the end users' perceived quality and extracting their expectations (i.e., the yellow box of ECT-diagram on the right) from their feedback (i.e., the yellow-arrow analysis activity). Furthermore, such evaluation and extracted knowledge shall be continuously applied within the evolution process where end users' satisfaction is tracked by continuously verifying the confirmation of their collective expectation (i.e., the loop). Thirdly, the thesis also contributes to the understanding of some important aspects of the context of software evolution using data-driven approaches (i.e., dotted yellow box in the middle).

# 3 RESEARCH DESIGN

The chapter presents the overall planning and structure of this thesis work as well as the brief introduction of the adopted research methods, the collected data, and the utilized analysis techniques. Specifically, Section 3.1 describes the structure of the research in terms of the mapping of publications towards the potential answers to the previously proposed research questions. Section 3.2 introduces case study as the main research method adopted across the publications. Section 3.3 briefly introduces the research data collected and used in the case studies, as well as the data mining, opinion mining and machine learning techniques that are adopted to analyze such collected data.

## 3.1 Research Structure

The main goal of the the research is to propose a data-driven approach that continuously extracts the needs and expectation of the majority of software end users and monitors their satisfaction through the software evolution phase with unique context information taken into account. Therefore, such an approach shall provide the features mentioned above including: 1) specification of perceived quality, 2) specification of needs and expectation, 3) confirmation analysis, and 4) context analysis. Such a four-fold research structure in then shown in Figure 3.1.

In Figure 3.1, the green round-cornered boxes depict the activities in the approach process, where the according techniques and/or methods adopted (marked as blue letters) are listed within as bullet-points. The gray rectangles are the inputs and/or outputs of the activities. The publications (marked in green letters) are the final outputs of the four specified critical contributions mentioned above and are also mapped accordingly. Furthermore, the contributions of the publications answering each of the previously proposed research questions are also shown in the figure. To be noted, the activity marked in gray indicates that the activity *situational context*

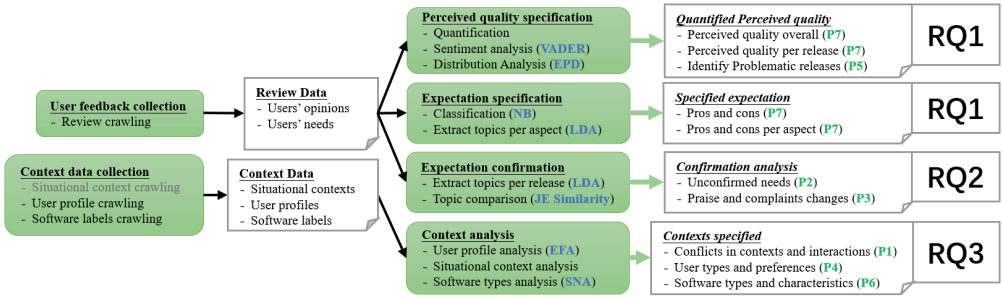


**Figure 3.1** The Research Framework Overview

*crawling*, due to the limitation of techniques and privacy issues, has not yet been covered in this thesis and shall be discussed as future work.

## 3.2 Case Study as Research Method

Within the publications, case study is the research method that has been applied in the majority. Case study research method is defined as "*an empirical inquiry that investigates a contemporary phenomenon within its real-life context; when the boundaries between phenomenon and context are not clearly evident; and in which multiple sources of evidence are used*" [192]. It closely examines the data within a specific context by selecting a limited number of individuals as subjects, which, in its essence, explores and investigates real-life phenomena through detailed contextual analysis of a limited number of events or conditions, and their relationships [193]. Though commonly used in psychology, sociology, political science and etc, case studies are also commonly applied in software engineering domain. The reason is that software engineering is largely to investigate how the proposed approaches are conducted by stakeholders under different conditions when case study offers an approach that does not need a strict boundary between such conditions and the objectives [156].

According to Runeson and Höst's study on the research purpose types, case study as research method can be classified into four types [156], including: *Exploratory*, *Descriptive*, *Explanatory*, and *Improving*. Based on the research questions proposed, it is reasonable to adopt the explanatory case studies as the method towards the verification of proposed data-driven approaches. As the answer to each research question (or sub-questions) is a data mining and analysis pipeline, the usefulness of such a pipeline shall be verified in the cases where it is applied with real-world datasets. Regarding

each proposed research question, the research is designed similarly. For example, to answer RQ1, the proposed data-driven method shall contain a set of steps for the extraction of users' collective expectation and a set of steps for the evaluation of the perceived quality in use. Hence, to verify such outcomes, the case study shall be designed with the proposed method applied on an example software product (or several software products) with its reviews collected. The case study shall follow exactly the steps proposed in the method with the outcome being the summarized users' expectation and needs (i.e., topics) as well as the quantified perceived quality in use (i.e., sentiment).

## 3.3 Data and Analysis Techniques

This subsection shall introduce briefly the data used in the publications and the methods of data gathering. It also provides an overview on the adopted analysis techniques.

### 3.3.1 Data Gathering

The main data used in this thesis is software end user textual reviews, which are obtained from online distribution platforms, including Google Play and Steam. Compared to the other review data sources, choosing these two platforms both provide API to ease the data retrieval efforts. Additionally, even targeting the ease of use for the official API of these platforms, useful open-source solutions are provided by adopting external web crawling tools, e.g., Node.js and Scrapy [73, 149]. On the other hand, these two platforms provide user reviews in different forms. For mobile applications on Google Play, the users tend to provide quick and short reviews; for video games on Steam, the players prefer longer text reviews with more details on the pros and cons. Specially, seen as hedonic information systems [180], digital games share certain common grounds with utilitarian software products, when the user reviews shall similarly reflects their opinions on the quality.

Specifically, the data used in the publications for case studies, as well as the retrieving methods and sources, are listed in Table 3.1. To be noted, Publication I proposes the reference model of situational context and user-app interaction analysis, where the data-driven approach and the case study are not used.

| Publication | Volume | Type | Source | Retrieving Method |
|---|---|---|---|---|
| II | 153128 | Text Review | Google Play | Google Play API |
| III | 1148032 | Text Review | Google Play | Google Play API |
| IV | 60267 | User Profile | Steam | BeautifulSoup |
| V | 3793125 | Text Review | Google Play | Google Play API |
| VI | 23034 | Game (with Tags) | Steam | Steam API & Scrapy |
| VII | 99993 | Text Review | Steam | Steam API & Scrapy |

**Table 3.1** Data Information for each Publication

### 3.3.2 Analysis Techniques

In order to answer the previously defined research questions, particular analysis techniques are applied across the publications. For example, sentiment analysis is applied in order to detect the user opinions from their reviews; topic modeling is used to specify the significant topics among the reviews that reflect users' needs; binary classification and multi-labeled classification are used to classify the user reviews based on pre-defined aspects. Hereby, each data analysis technique that has been applied in the publications is introduced as follows.

**Sentiment Analysis with VADER.**
*(Publication: II, III, V, VII; Research Question: RQ1, RQ2)*

In general, sentiment analysis is to detect the polarity (e.g. a positive or negative opinion) within the given text, regardless the length of it. The aim is to measure the attitude, sentiments, evaluations, attitudes, and emotions of a speaker/writer based on the computational treatment of subjectivity in a text [113].

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media [65]. VADER uses a combination of A sentiment lexicon is a list of lexical features (e.g., words) which are generally labeled according to their semantic orientation as either positive or negative. VADER not only tells about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is.

The lexicon for sentiment analysis is a list of words used in English language, each of which is assigned with a sentiment value in terms of its sentiment valence (intensity) and polarity (positive/negative). Therefore, as each text can be seen as a list of

words, a lexicon is selected to determine the sentiment score of each word. Furthermore, a rational value within a range is assigned to a word. For example, if the word "okay" has a positive valence value of 0.9, the word "good" must have a higher positive value, e.g., 1.9, and the word "great" has even higher value, e.g., 3.1. Furthermore, the lexicon set shall include commonly-adopted social media terms, such as Western-style emoticons (e.g., :-)), sentiment-related acronyms and initialisms (e.g., LOL, WTF), and commonly used slang with sentiment value (e.g., nah, meh).

With the well-established lexicon, and a selected set of proper grammatical and syntactical heuristics, the overall sentiment score of text can be determined. The grammatical and syntactical heuristics are seen as the cues to change the sentiment of word sets. Therein, punctuation, capitalization, degree modifier, and contrastive conjunctions are all taken into account. For example, the sentiment of "The book is EXTREMELY AWESOME!!!" is stronger than "The book is extremely awesome", which is stronger than "The book is very good.".

**Topic Modeling with LDA.**

*(Publication: II, III, V, VII; Research Question: RQ1, RQ2)*

Topic modeling is a commonly adopted unsupervised method to classify documents in order to detect the latent set of topics. It is for automatically summarizing a large volume of text archives where the outcome can be used to facilitate the understanding of such texts. Therein, Latent Dirichlet Allocation (LDA) is a commonly adopted standard tool in topic modeling [11].

LDA is a generative probabilistic model of a corpus. The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words [11]. The following assumptions are commonly considered as the pre-conditions when building an LDA topic model:

- Every piece of text is seen as a collection of words (i.e., "bag of words") when the order and the grammatical role of these words are irrelevant in terms of the topic model.
- Stopwords, e.g., "are", "but", "the", etc. can be eliminated during preprocessing due to the fact that very limited useful information is carried therein regarding the topics.
- Words that are commonly used in majority of the texts (e.g., 80% 90%) are also irrelevant to the topics. They can be eliminated as well.

- The number of topic, i.e., $k$, is pre-defined.
- When assigning any particular word to a topic, the assumption is all the previous topic assignments are correct. Therefore, in this way, iteratively updating the assignment of from word to word using the model shall then create the documents.

In general, based on the known "word-document" belonging relations, LDA aims to calculate how likely each word belongs to a particular topic. Therefore, the core process of building the LDA topic model is as follows:

1. Randomly assign each word in the documents to one of $k$ topics.
2. Go through each word $w$ in each document $d$, and calculate:
    - The proportion of words in document $d$ that are assigned to topic $t$.
    - The proportion of assignments to topic $t$ over all documents that come from this word $w$.
3. Update the probability for the word $w$ belonging to topic $t$, as $P(t|d) \times P(w|t)$.

On the other hand, in order to find the best topic number, the topic coherence that represents the quality of the topic models is commonly applied. Topic coherence measures the degree of semantic similarity between high scoring words in the topic. A high coherence score for a topic model indicates the detected topics are more interpretable. Thus, by finding the highest topic coherence score, the most fitting topic number can be determined. For example, in Publication VII, $c\_v$ coherence measure is used to detect the best fitting topic number. It is based on a sliding window, one-set segmentation of the top words and an indirect confirmation measure that uses normalized pointwise mutual information (NPMI) and the cosine similarity [175]. Normally, the model that has the highest $c\_v$ value before flattening out or a major drop shall be selected in order to prevent the model from over-fitting.

**Topic Similarity Analysis.**
*(Publication: II, III; Research Question: RQ2)*

As each topic is represented as a set of keywords, the similarity of two topics shall be denoted by the common keywords of these topics. Hence, an easy way for calculating the similarity between any two topics (e.g., $t_i$ and $t_j$) is by using the

Jaccard similarity. It reflects the percentage of the common keywords of the two sets in the whole keywords set of the two: $J(t_i, t_j) = \frac{|t_i \cap t_j|}{|t_i \cup t_j|}$. However, by using the Jaccard Similarity, we consider two given topics are similar only when they contain a particular number of common keywords, regardless of the probability of them. The meaning of each topic shall be more likely reflected by the high-probability keywords of the topic. Furthermore, the subset of only low-probability keywords may reflect different meanings.

Hence, when comparing the similarity of two given topics, the probability of the common keywords shall be taken into account. Considering that Jaccard coefficient is the normalized inner product [177], two potential similarity measure methods are adopted: the Kumar-Hassebrook (KH) similarity [89] and the Jaccard Extended (JE) Similarity. Provided between topic $t_i$ and $t_j$, the $c$ common keywords are denoted as $[kw_{ij,1}, kw_{ij,2}, ...kw_{ij,c}]$, with the according probability list in $t_i$ and $t_j$ is $[p_{i,1}, p_{i,2}, ...p_{i,c}]$ and $[p_{j,1}, p_{j,2}, ...p_{j,c}]$. The similarity of the two given topics by the two similarity calculation methods are described respectively as follows.

- Kumar-Hassebrook (KH) similarity:

$$KH(t_i, t_j) = \frac{\sum_{x=1}^{c} p_{i,x} \cdot p_{j,x}}{\sum_{x=1}^{k} p_{i,x}^2 + \sum_{x=1}^{k} p_{j,x}^2 - \sum_{x=1}^{c} p_{i,x} \cdot p_{j,x}} \tag{3.1}$$

- Jaccard Extended (JE) Similarity:

$$JE(t_i, t_j) = \frac{\sum_{x=1}^{c} \frac{p_{i,x}+p_{j,x}}{2}}{\sum_{x=1}^{c} \frac{p_{i,x}+p_{j,x}}{2} + \sum_{x=c+1}^{k} p_{i,x} + \sum_{x=c+1}^{k} p_{j,x}} \tag{3.2}$$

The probability for each keyword of any topic belongs to (0,1). Hence, for this formula, when $t_i$ and $t_j$ contain more common keywords, the numerator increases monotonically, and the denominator decreases monotonically. Therefore, $KH(t_i, t_j)$ and $JE(t_i, t_j)$ both increase when $t_i$ and $t_j$ have more keywords in common. In addition, when the probability of the common keywords increases, $\sum_{x=1}^{c} p_{i,x} \cdot p_{j,x}$ and $\sum_{x=1}^{c} \frac{p_{i,x}+p_{j,x}}{2}$ increase as well. Because the denominator is greater than the numerator, and both are greater than 0, both $KH(t_i, t_j)$ and $JE(t_i, t_j)$ increases when the probabilities of the common keywords of $t_i$ and $t_j$ increase. Hence, either KH or JE similarity can be used to calculate the similarity of given topics by taking into

account both the number of common keywords and the probabilities of such keywords.

**Naive Bayes Classification.**

*(Publication: II, III, VII; Research Question: RQ1, RQ2)*

Naive Bayes (NB) classifier is a easy-to-use model that's commonly applied towards typical classification problems, such as, herein the classification of textual reviews. The foundation of the classifier being Bayes' Theorem works towards the computation of the conditional probability as follows,

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

where,

- $P(A|B)$: the probability of event $A$ occurring when event $B$ has occurred.
- $P(B|A)$: the probability of event $B$ occurring when event $A$ has occurred.
- $P(A)$: the probability of event $A$ occurring
- $P(B)$: the probability of event $B$ occurring.

When applying Bayes' theorem, the "naive" assumption lies where features are independent of each other with no correlation in between. When regarding the text as data, the independence assumption lies between the individual words in each piece of text. Thus, the Bayes' theorem for multi-feature variables can be described as, given class variable $y$ and dependent feature vector $[x_1, x_2, ..., x_n]$,

$$P(y|x_1, x_2, ..., x_n) = \frac{P(x_1, x_2, ..., x_n|y) \times P(y)}{P(x_1, x_2, ..., x_n)}$$

Therefore, with the "naive" assumption taken into account,

$$P(x_i|y, x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_n) = p(x_i|y)$$

For all $i$,

$$P(y|x_1, x_2, ..., x_n) = \frac{\prod_{i=1}^{n} P(x_i|y) \times P(y)}{P(x_1, x_2, ..., x_n)}$$

Due to the fact that $P(x_1, x_2, ..., x_n)$ can be calculated when the texts entered as training data is determined, the classification shall follow,

$$P(y|x_1, x_2, ..., x_n) \propto \prod_{i=1}^{n} P(x_i|y) \times P(y)$$

$$\hat{y} = \arg\max_{y} \prod_{i=1}^{n} P(x_i|y) \times P(y)$$

To be noted, the different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i|y)$. For example, for Gaussian Naive Bayes classification, such likelihood of the features is assumed as follows.

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

**Exploratory Factor Analysis.**

*(Publication: IV; Research Question: RQ3)*

Exploratory factor analysis (EFA) [58] is used to detect the latent variables sharing common variances within the user profile data [6]. These factors that are detected with EFA are the hypothetical constructs to represent variables, which cannot be directly measured [21]. The method discovers the number of factors and the combinations of measurable variables that influence each individual factor [34]. EFA can reduce the complexity of the data and simplify the observations with smaller set of latent factors as well as the relations between variables. Meanwhile, parallel analysis (PA) [63] is to detect the proper number of factors. In PA, the Monte Carlo simulation technique is employed to simulate random samples consisting of uncorrelated variables that parallel the number of samples and variables in the observed data [63]. From each such simulation, eigenvalues of the correlation matrix of the simulated data are extracted, and the eigenvalues are averaged across several simulations [63]. The eigenvalues extracted from the correlation matrix of the observed data, ordered by magnitude, are then compared to the average simulated eigenvalues, also ordered by magnitude. The decision criteria is that the factors with observed eigenvalues higher than the corresponding simulated eigenvalues are considered significant. To simplify interpretation of the factor analysis result, the *varimax* rotation technique [75] is normally employed to maximize the variance of the each factor loading.

**Social Network Analysis.**

*(Publication: VI; Research Question: RQ3)*

Two classic centrality measures: closeness centrality and betweenness centrality [44], are commonly applied to the network of software labels to analyze the important software characteristics. In addition, PageRank, a popular algorithm measuring the importance of website pages [16], is also adopted herein to measure the importance of label vertices, compared with the results of the centrality measures. On the other hand, Louvain method for community detection, a method to extract communities from large networks [12], is used to obtain the latent communities of the software label network.

**Closeness and Betweenness Centrality.**

Closeness centrality of a network is to measure the steps of information to travel from one vertex to the others [136]. For a network with $V$ vertices set, the closeness centrality of any vertex $i$ in the network, $C_C(i)$, is calculated as

$$C_C(i) = \frac{1}{\sum_{j \in V} D_G(i,j)} \tag{3.3}$$

where $d_G(i,j)$ is geodesic distance (i.e., the shortest path) between $i$ and another vertex $j \in V$.

Betweenness centrality, on the other hand, measures the shortest paths through a particular vertex. It is the "brokering positions between others that provides opportunity to intercept or influence their communication" [15]. The betweenness centrality of vertex $i$, $C_B(i)$, is calculated as

$$C_B(i) = \sum_{j,h \neq i} \frac{g_{hij}}{g_{hj}} \tag{3.4}$$

where $g_{hij}$ is the number of geodesic paths between another two vertices $h, j \in V$ through vertex $i$.

**PageRank.** PageRank assigns universal ranks to web pages based on a weight-propagation algorithm, which is the core of Google search engine [16]. In general, for PageRank, if the sum of the backlink ranks for a page is high, it is ranked high. For a page $w$ with $k$ links, the PageRank value of $w$ is computed as

$$PR(w) = \frac{1-d}{N} + d \sum_{i=1}^{k} \frac{PR(w_i)}{C(w_i)} \tag{3.5}$$

where,

- $d \in (0,1)$ is the probability of the user goes to random page instead of going to one of the links within.
- $C(w)$ is the number of links going out of $w$
- $N$ is the total number of pages.

**Network Modularity and Community Detection** A group of vertices of a network that have denser connections amongst one another than those with the other vertices form a network community [49, 151]. Thus, community detection is necessary for the identification of such communities of a particular network so that the structure of it can be revealed. Meanwhile, the modularity of a network is a measure indicating the strength of division of a network into communities [130]. To simply put, networks with higher modularity are stronger connected within the vertices within.

The modularity $Q$ of a network with the set of vertices $V$ and the set of edges $E$ can be computed as

$$Q = \sum_{k=1}^{m} [\frac{l_k}{|E|} - (\frac{d_k}{2|E|})^2] \tag{3.6}$$

where,

- $m$ is the community number
- $l_k$ is the number of edges between any two vertices from the $k$-th community
- $d_k$ is the sum of degree of all those vertices

Louvain community detection method is commonly used herein for the structure extraction of a large weighted network with optimized modularity value [12]. Each vertex is assigned to a community when $Q$ is maximized. $\Delta Q$ indicating the increased value of $Q$ when moving vertex $i$ to community $C$, which is calculated as

$$\Delta Q = \frac{\sum_C + k_i^C}{2n} - (\frac{\sum_{\widehat{C}+k_i}}{2n})^2 - [\frac{\sum_C}{2n} - (\frac{\sum_{\widehat{C}}}{2n})^2 - \frac{k_i}{2n}] \tag{3.7}$$

where,

- $k_i$ is the sum of weighted edges incident to $i$
- $k_i^C$ is the sum of the edges from $i$ to vertices in community $C$

- $\sum_C$ is sum of the weighted edges in $C$
- $\sum_{\widehat{C}}$ is the sum of the edges incident to vertices in $C$.
- $n$ is the sum of the weights of all the edges of the network.

The method is to detect the optimized community structure of a network by moving vertices from one community to another in order to detect the significantly improved $\Delta Q$ [32].

# 4 RESULTS

This chapter describes in details the results of the research. Section 4.1 presents the approach of evaluating the perceived quality of software product based on sentiment analysis of the user reviews. Section 4.2 introduces the approach of using topic modeling on user reviews to specify the expectations and needs of the end users. These two sections are the results drawn from Publication VII, which shall answer RQ1. Section 4.3 presents the enrichment of the quantified perceived quality evaluation approach proposed in Section 4.1 by providing a mechanism to detect the abnormal updates within the evolution process.This is the results drawn from Publication V. Section 4.4 presents the approach of monitoring end users' expectation and needs that are not confirmed and the changes of through the software evaluation timeline. These results are drawn from Publication II and III, which shall answer RQ2. Furthermore, Section 4.5 summarizes the approaches on contexts analysis that supports the above methods, including: 1) situational context analysis, 2) user types and preferences analysis, and 3) software genres and characteristics analysis. These results are drawn respectively from Publication I, IV and VI, which collectively answer RQ3. Additionally, Section 4.6 summarizes each selected publication.

## 4.1 Perceived Quality Analysis

The approach of perceived quality analysis starts with collecting end user reviews from online platforms (e.g., Google Play, Steam, etc.) via either API or web crawling. Then the obtained raw review data shall be preprocessed into structured form. The second step is to filter out the "non-informative" reviews via a pre-trained classifier. With the obtained informative reviews dataset, the third step is to classify the data into different perspectives according to a selected framework. The selected framework can be either generic, e.g., ISO/IEC 25010 [71], or more specific towards particular software category, e.g., playability framework for video games [141]. With

each review instance categorized into a specific perspective, the fourth step is to quantify the evaluation result of each perspective. Subsequently, the fifth step is to visualize such a result and present an intuitive summary.

**Step 1. Preprocessing.** The preprocessing step encompasses the following key activities. First, each review item is divided from the dataset into sentence-level review instances (e.g., with NLTK), due to the fact that each review with multiple sentences can contain multiple topics and various sentiments. Based on the obtained sentence-level review dataset, the second step is to build the bigram and trigram models to identify the phrases within the data (e.g., with Gensim). Subsequently, for each review sentence, a series of text processing activities are conducted, including transforming text into lowercase, removing non-alpha-numeric symbols, screening stop-words, eliminating extra white spaces, and lemmatization (e.g, with NLTK WordNetLemmatizer). Note that the processing is only applied to the text when topic modeling is required. For sentiment analysis, such activities are not only unnecessary but also counter-productive. The details of the process are explained as follows.

**Step 2. Filtering.** Herein, the filtering step is to classify the dataset of sentence-level review instances into *informative* and *non-informative*. It aims to identify the review sentences that contain description regarding the software perceived quality and screen out those not relevant. To efficiently identify and filter the non-informative review sentences, a pre-trained text classifier shall be applied hereby, e.g., the Naive Bayes (NB) classifier or the Expectation Maximization for Naive Bayes (EMNB) [133]. These algorithms are recommended due to the requirements of less effort in training data preparation and the satisfactory performance.

**Step 3. Classification.** This step aims to classify the obtained informative review sentences into perspectives according to the selected software quality framework. Targeting such classification objectives, when the classes are determined by the existing framework, a supervised learning algorithm is more suitable. To be noted, due to the common situation where a particular review sentence contains information regarding multiple perspectives, a multi-label classification algorithm, e.g., kNN classification method adapted for multi-label classification (MLkNN) [195], can be used instead of the traditional NB. To determine which method is better fit for the specific dataset, testing on classification accuracy is recommended.

**Step 4. Quantification.** With the classified three sets of informative review

sentences, this step is to evaluate each of the perceived quality perspectives by quantifying the overall opinions extracted from the according set of review sentences. Herein, the average sentiment score of the informative review sentences is used to represent the users' collective evaluation towards the perceived quality of the target software product.

**Step 5. Visualization.** The output of the quantification of end users' opinions regarding each perceived quality perspective can be visualized with a radar diagram. For example, a triangular radar chart can be used to display the quantification results based on a 3-perspective framework.

To validate the proposed perceived quality analysis approach, Publication VII conducts a case study analyzing the perceived playability (i.e., video game quality [141]) of a particular video game product, No Man's Sky[1]. The reason of choosing this case is due to the fact that this game had suffered from a disaster first release and have been constantly maintained and updated by the developers. Therefore, such a "ground true" can be used as validation to the case study outcomes. For this case study, 99993 English reviews from 2016-08-12 to 2020-06-07 are collected with 519,195 review sentences obtained via tokenization. Using the trained EMNB classifier (F1-score: 0.85), 273476 informative review sentences are obtained. Paavilainen's playability framework [141] is adopted as the pre-defined class set, where *functionality*, *gameplay*, and *usability* are the three perspectives for the classification. Using MLkNN (accuracy: 0.769), the 273476 informative review sentences are classified into {Functionality: 43110, Gameplay: 205474, Usability: 30176}. Furthermore, using the VADER sentiment analysis approach [65], the average sentiment scores for the sentences classified to each perspective are: {Functionality: 0.025, Gameplay: 0.111, Usability: 0.039} which indicates the mediocre level of overall quality, which can be validated by the Steam overall review of "Mixed". The according visualization of the previously quantified results is shown in Figure 4.1.

[1] https://www.nomanssky.com/

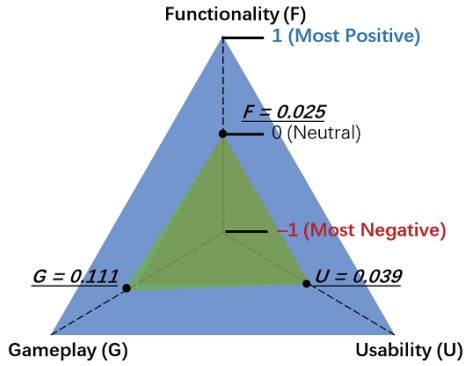


**Figure 4.1** Visualization of the Case Study.

The results show that the proposed approach can effectively analyze and evaluate the perceived quality of a particular software product based on the reviews of its end users. Such evaluation can also be visualized.

## 4.2 Expectation and Needs Analysis

The initial steps of the approach for expectation and needs analysis, i.e., preprocessing, filtering and classification steps, are identical to the previous perceived quality analysis, due to the fact that informative and classified review sentences sets is the input. The unique steps for the approach are as follows.

**Step 1. Divide by Sentiment.** The obtained informative review sentences for each perspective (after the classification step) are divided into two subsets, i.e., positive and negative, based on their sentiment. To be noted, sentences with neutral sentiment shall be ignored.

**Step 2. Determine topic numbers.** For each subset of sentences, the most suitable topic number shall be determined using a proper method. For example, Publication VII adopts the $c\_v$ coherence measure, which is based on a sliding window, one-set segmentation of the top words and an indirect confirmation measure that uses normalized pointwise mutual information (NPMI) and the cosine similarity [175].

**Step 3. Extract Topics.** Based on the previously determined topic numbers, the topics are extracted from each subset of review sentences that are previously classified. The interpretation of these topics shall reflect the expectation and needs of the end users in terms of the particular perspective.

The case study conducted in Publication VII also validates the proposed user expectation and needs analysis approach using the review data from No Man's Sky. The 273476 informative review sentences are classified into the three perspectives with each perspective further classified by the positive and negative sentiment. Furthermore, for each subset of text, the proper topic number is determined using $c\_v$ coherence measure. Based on the determined topic numbers, for each subset of review sentences, the topics are extracted and interpreted as Table 4.1.

| Topic (Positive Functionality) | Top Keywords |
|---|---|
| + Load Screen and Crashing | "game","play","crash","time","screen","start","hour","go","load","get" |
| + Performance and bugs fixed via update | "issue","game","performance","fix","people","update","would","bug","problem","patch" |
| + Running game fine with settings | "run","game","setting","graphic","work","get","fps","pc","fine","high" |
| **Topic (Negative Functionality)** | **Top Keywords** |
| − Poor Performance, Bugs, Crash, Need Fix | "game","issue","problem","performance","fix","people","crash","bad","poor","bug" |
| − Lag, Stutter, fps drop, even with low settings | "run","setting","low","game","stutter","drop","pc","graphic","lag","fps" |
| − Crash at Start screen, try hours | "crash","game","play","time","screen","get","start","can","try","hour" |
| **Topic (Positive Gameplay)** | **Top Keywords** |
| + Explore, survival, different planet systems | "planet","find","new","explore","system","beautiful","different","look","survival","thing" |
| + Crafting, ship-flying, resource and inventory | "space","ship","get","resource","fly","well","craft","upgrade","inventory","learn" |
| + Fun exploration gameplay | "game","exploration","fun","play","get","hour","gameplay","good","enjoy","lot" |
| + Need story to make better | "game","want","make","need","would","give","bit","people","story","work" |
| **Topic (Negative Gameplay)** | **Top Keywords** |
| − Repetitive, boring gameplay | "game","get","hour","feel","repetitive","start","bore","boring","gameplay","people" |
| − Lack of inventory upgrade | "ship","resource","make","need","inventory","find","upgrade","lack","craft","much" |
| − Fly, explore, combat | "planet","space","see","look","explore","combat","find","fly","kill","ship" |
| **Topic (Positive Usability)** | **Top Keywords** |
| + Control feels with controller, fly ship | "control","use","ship","take","feel","get","controller","fly","space","flight" |
| + Beautiful graphics | "game","graphic","play","change","setting","beautiful","look","run","work","good" |
| + Music&sound, hold and click button | "hold","button","music","menu","screen","system","inventory","click","sound","second" |
| **Topic (Negative Usability)** | **Top Keywords** |
| − Graphic settings poor, restart | "graphic","game","setting","change","run","bad","start","poor","get","restart" |
| − Fly control with mouse annoying | "control","mouse","ship","fly","game","use","get","annoying","make","press" |
| − Terrible texture and sound | "terrible","look","texture","game","sound","pop","point","require","complaint","way" |
| − Horrible flight control, cluncky inventory | "control","flight","feel","people","horrible","system","inventory","lack","clunky","fov" |
| − Option, click and hold button, bad/awful PC port | "game","pc","option","button","hold","port","menu","awful","bad","click" |

**Table 4.1** Extracted Topics for Each Review Set

Then the extracted complaints and praises from the six sub sets of reviews are compared with the expert opinions extracted from the critic reviews of Metacritic[2] that provide explicit pros and cons. It shows that a great majority of the pros and

[2]https://www.metacritic.com/game/pc/no-mans-sky

cons across all three perspectives mentioned by the media experts are also reflected by the topics extracted from the reviews. Therefore, this case study also validates the effectiveness of the expectation and needs analysis approach.

## 4.3 Abnormal Update Detection

The goal of this step of approach is to detect the time span that contains an abnormal amount of negativity reflected by the user reviews. It is reasonable to presume that the changes in the users' collective opinions and those of the topics of the reviews are correlated. In addition, the software update that most likely causes such negativity as well as the changes in topics and sentiments can also be identified by the similarities between the update description texts and the according reviews. This approach contains three steps: 1) detect the distribution of the reviews sentiment changes, 2) identify the abnormal sentiment changes, and 3) identify the abnormal update that is connected to the previously identified anomaly.

**Step 1. Detect Sentiment Change Distribution.** Herein, the sentiment change can be defined as the increase or decrease of the average sentiment score of the review texts compared to the former time period. It is also possible to define it as the increase or decrease rate of negative review numbers. In this way, the strength of sentiment can be ignored. Ideally, such sentiment change data shall fit a distribution model where majority of the values are close to 0 with the rest evenly distributed on both sides. The Kolmogorov–Smirnov test (K-S test) [117] is applied to detect the distribution of the data. Among 87 different distribution models[3], the sentiment change data of five mobile apps in one year fits Generalized Normal Distribution (Version 1. Exponential Power Distribution) [127] with a $p$ value of 0.968 ($\mu = 0.0002$, $\alpha = 0.0227$, $\beta = 1.0444$).

**Step 2. Identify Abnormal Sentiment Changes.** Based on the distribution model detected that fits the sentiment change data, the abnormal sentiment changes can be identified by the $3-\sigma$ rules. For example, with the exponential power distribution model, 1,000,000 samples are simulated having such distribution with the above-obtained parameters. The calculated number of samples lying in the band around the mean with $\pm 3\sigma$ width is 986,424, indicating 98.6% values shall be considered normal. Thus, any values that are placed out of the 98.6% confidence inter-

[3] https://docs.scipy.org/doc/scipy/reference/stats.html

vals are considered "abnormal". According to Chebyshev's inequality [22], no more than $1/k^2$ of the distribution's values can be more than $k$ standard deviations away from the mean. Thus, when $1 - 1/k^2 = 0.986$, the values that are around $k = 8$ standard deviations away from the mean are abnormal.

**Step 3. Identify Problematic Updates.**

To detect the connections between reviews with abnormal negativity and the descriptions of the updates, Word2Vec model [123] is applied. The similarity between the update descriptions and the reviews can be measured by the similarities of those words from both parties with high TF-IDF [153]. It is likely to identify the nearest update causing the anomaly in review sentiment changes.

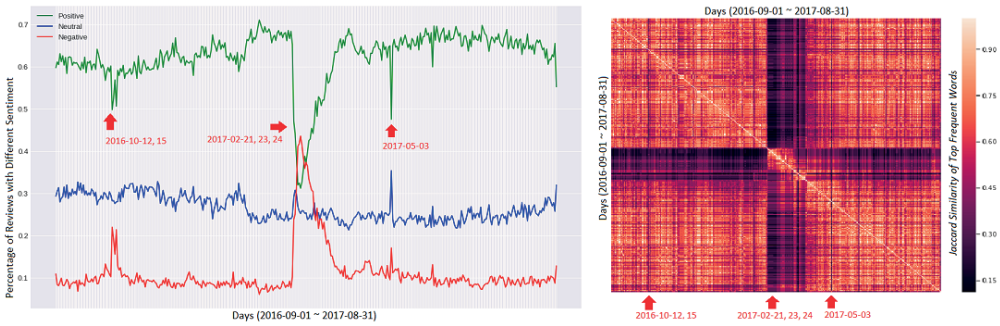


**Figure 4.2** The Identified Abnormal Days of Whatsapp as an Example

A case study is conducted in Publication V to verify the approach using the user reviews of five instant messenger mobile apps from Android platform, namely *Imo*, *Hangouts*, *Messenger*, *Skype* and *Whatsapp*. For each app, the reviews from 2016-09-01 to 2017-08-31 are collected with 2676488 review sentences. With the 1840 daily sentiment change values, the best fitting distribution model detected is the Exponential Power Distribution (EPD). Taking Whatsapp as an example, the six abnormal days can be verified by the obvious rise in negative sentiment and fall of positive sentiment, as well as the sudden changes in top frequent words (shown in Figure 4.2). the abnormal days identified can be mapped to major updates by comparing the similarity between the negative review content of the identified abnormal days and the description text of the major updates. Still taking Whatsapp as an example, Figure 4.3 shows the similarity of the negative reviews of each identified abnormal day and the descriptions of the 9 major updates. Therein, the nearest previous update of each abnormal day is marked red. Show in the figure, for the identified abnormal days of Whatsapp: 2017-02-21, 2017-02-23, 2017-02-24 and 2017-05-03, the similarities of

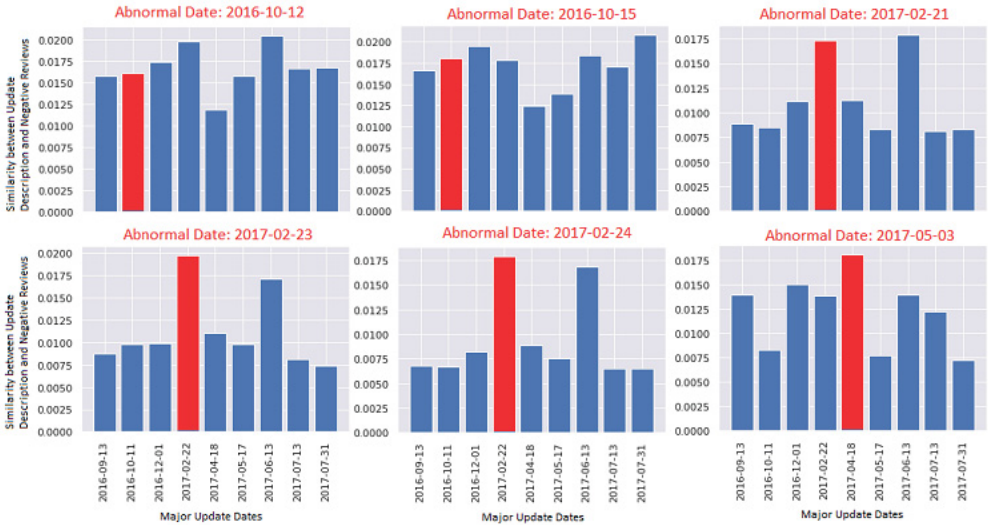


**Figure 4.3** Similarity between Update Description and Negative Reviews of Abnormal Days for Whatsapp

the negative reviews and the descriptions of their nearest previous updates are significantly high.

## 4.4 Software Evolution Monitoring

In this thesis, the core of the software evolution monitoring is to keep track on whether each of the software updates addresses the users' expectations and complaints previously. Therefore, it is compulsory to firstly understand, for each particular update, the topic changes before and after it. Ideally, given a particular software update $u_i$, any review $r \in R_i$ provided after the release time of $u_i$ and before that of $u_{i+1}$ is seen as the ones regarding only $u_i$. Thus, given $T_i^+$ and $T_i^-$ as the topic set for the positive and negative reviews of $R_i$, by comparing the similarities and changes between $T_{i-1}^+$, $T_{i-1}^-$, $T_i^+$, and $T_i^-$, the following hypotheses are proposed and can be verified.

- H1. The topic similarities between $T_{i-1}^+$ and $T_i^+$ reflect the *merits* regarding the software in general.
- H2. The topic similarities between $T_{i-1}^+$ and $T_i^-$ reflect the *uncomfortable changes* in the update $u_i$.

- H3. The topic similarities between $T_{i-1}^{-}$ and $T_i^{+}$ reflect the *improvement* in the update $u_i$.
- H4. The topic similarities between $T_{i-1}^{-}$ and $T_i^{-}$ reflect the *remaining issues* regarding the software.

In order to identify the matching topics between two review topic sets $T_a$ and $T_b$, the aim is to identify all the topic pairs $(t_{ai}, t_{bj})$, $t_{ai} \in T_a$ and $t_{bj} \in T_b$, that have the high similarity. To calculate the similarity value between topics, either JE or KH similarity is used, where a threshold to determine the matching topics is also needed (e.g., Publication II set the the threshold as the original Jaccard similarity value).

Furthermore, to extend such analysis regarding a particular update to the software evolution timeline, we shall be able to observe the changes of users' expectation and needs within such a longer period of time. Therefore, given a set of updates $U = \{u_1, u_2, \ldots u_n\}$ with the according review sets $R_1^{+}, R_1^{-}, R_2^{+}, R_2^{-}, \ldots R_n^{+}, R_n^{-}$, finding the topic similarity correlations throughout $T_1^{+}$, $T_1^{-}$, $T_2^{+}$, $T_2^{-}$, $\ldots$ $T_n^{+}$, $T_n^{-}$ shall reflect the evolving user opinions concerning various aspects of the app within a particular period of time. Therefore, provided a set of topic $\{t_1, t_2, \ldots t_i \ldots t_n | t_i \in T_i\}$, where each $t_i$ is similar to $t_{i+1}$ $(1 \leq i < n)$, then such topic set is defined as a *similar topic chain*. Three additional hypotheses are proposed and verified as follows.

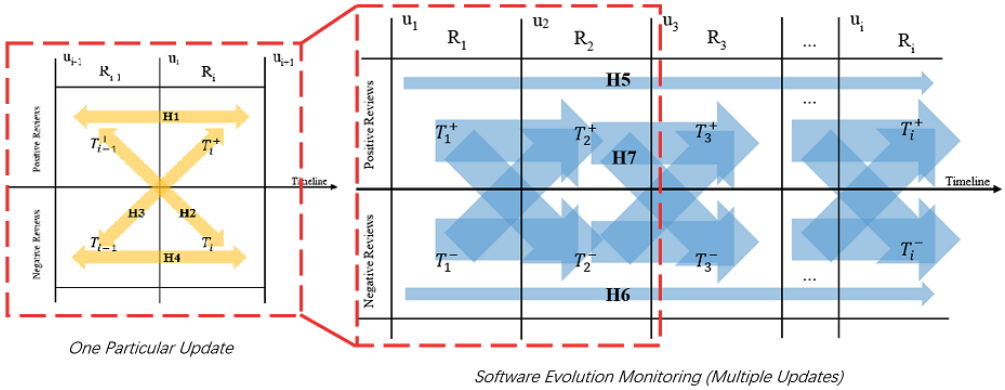


**Figure 4.4** Hypotheses on Software Evolution Monitoring Mechanism (adapted from Publication II and III)

- H5. The similar topic chains through $T_i^{+}$, $T_{i+1}^{+}$, $\ldots$ $T_n^{+}$ reflect the merits and users' praise regarding a sequence of software's updates $u_i$, $u_{i+1}$, $\ldots$ $u_n$.
- H6. The similar topic chains through $T_i^{-}$, $T_{i+1}^{-}$, $\ldots$ $T_n^{-}$ reflect the issues and

users' complaints regarding a sequence of software's updates $u_i$, $u_{i+1}$, ... $u_n$.

- H7. The similar topic chains containing topics from both $T_i^+$, $T_{i+1}^+$, ... $T_n^+$ and $T_i^-$, $T_{i+1}^-$, ... $T_n^-$ reflect the changing user opinions (positive to negative, or negative to positive) regarding particular aspects of the software.

Figure 4.4 shows the above proposed hypotheses in terms of the reflecting of changes in users' opinions via topic comparison amongst different review sets. To validate the hypothesis, case studies in Publication II and III use the end user reviews of *Skype*[4] on Google Play from 1.9.2016 to 31.8.2017 and those of *Whatsapp*[5] between 2016-09-13 and 2017-08-31. Respectively, 153128 and 1148032 reviews are collected.

The major update version-1.6.2017 of Skype from Google Play is selected, due to the fact that this update provides significant changes in the UI design and new user experiences. Therein, the similar topic pairs between the according $T_{i-1}^+$, $T_{i-1}^-$, $T_i^+$, and $T_i^-$ are obtained with the KH similarity values. The similar topic pairs identified are shown in Table 4.2.

these detected similar topics across the review sets can be verified by the short notes of Skype developers regarding their updates. Specifically, the above topics can be associated with the following original developers claims:

- General performance and reliability improvements (Version 2017.08.15, Version 2017.08.29: phone calls, video calls and messaging quality)
- Improved sign in - sign back into your account more easily (Version 2017.08.15: "log in" features, user account related functions)
- New controls added to help users manage vibration and LED notification alerts. (Version 2017.07.05: notification),
- Improvements to PSTN call stability (Version 2017.07.05: connection)
- Messaging improvements – Add content to chats via the + button and enjoy more room for your messages. (Version 2017.08.02: messaging)
- The ability to add or remove contacts from your profile (Version 2017.08.01)
- Activity indicators - see who's currently active in your Chats list (Version 2017.08.02: contacts and statuses).

[4]https://play.google.com/store/apps/details?id=com.skype.raider
[5]https://www.whatsapp.com/

| **The Merits in General (Positive Before Update - Positive after Update)** | | |
|---|---|---|
| Topic Pair | Interpretation | Common Keywords |
| $(t^{+}_{(i-1)8}, t^{+}_{i5})$ | calling feature works | ['call', 'phone', 'sound', 'work'] |
| $(t^{+}_{(i-1)6}, t^{+}_{i6})$ | chat in group with video and calls | ['call', 'chat', 'friend', 'group', 'hear', 'make', 'people', 'person', 'phone', 'see', 'video'] |
| $(t^{+}_{(i-1)1}, t^{+}_{i7})$ | connect with family and friends | ['application', 'communicate', 'connect', 'family', 'friend', 'get', 'help', 'touch', 'way'] |
| $(t^{+}_{(i-1)7}, t^{+}_{i9})$ | add features and bugs fixed | ['add', 'bug', 'everything', 'fix', 'hope', 'issue', 'make', 'need', 'please'] |
| $(t^{+}_{(i-1)6}, t^{+}_{i5})$ | the video and sound quality | ['call', 'make', 'phone', 'quality', 'video', 'voice'] |
| **The Uncomfortable Changes (Positive Before Update - Negative after Update)** | | |
| Topic Pair | Interpretation | Common Keywords |
| $(t^{+}_{(i-1)6}, t^{-}_{i10})$ | interface, connection, and calling quality | ['call', 'connect', 'hear', 'make', 'person', 'phone', 'quality', 'video', 'voice'] |
| $(t^{+}_{(i-1)5}, t^{-}_{i6})$ | accounts, login, sign up, passwords | ['account', 'go', 'keep', 'let', 'log', 'password', 'sign', 'try', 'win'] |
| $(t^{+}_{(i-1)7}, t^{-}_{i8})$ | bugs fixes | ['fix', 'please', 'problem', 'thing'] |
| $(t^{+}_{(i-1)8}, t^{-}_{i10})$ | call connection drops, sounds | ['call', 'connection', 'drop', 'phone', 'sound', 'work'] |
| **The Improvement (Negative Before Update - Positive after Update)** | | |
| Topic Pair | Interpretation | Common Keywords |
| $(t^{-}_{(i-1)7}, t^{+}_{i3})$ | update in general | ['get', 'update', 'win', 'work'] |
| $(t^{-}_{(i-1)10}, t^{+}_{i8})$ | message notification | ['message', 'notification', 'open', 'see', 'send', 'show', 'take'] |
| $(t^{-}_{(i-1)3}, t^{+}_{i6})$ | calling | ['call', 'get', 'hear', 'make', 'people', 'person', 'phone', 'see', 'talk', 'time', 'video'] |
| $((t^{-}_{(i-1)3}, t^{+}_{i5})$ | video and sound quality | ['call', 'make', 'phone', 'quality', 'sound', 'time', 'video', 'voice'] |
| **The Remaining Issues (Negative Before Update - Negative after Update)** | | |
| Topic Pair | Interpretation | Common Keywords |
| $(t^{-}_{(i-1)3}, t^{-}_{i10})$ | update in general | ['call', 'connect', 'drop', 'hear', 'make', 'person', 'phone', 'quality', 'sound', 'time', 'video', 'voice'] |
| $(t^{-}_{(i-1)10}, t^{-}_{i3})$ | send and get messages with notifications | ['get', 'message', 'notification', 'open', 'see', 'send', 'show', 'take', 'time'] |
| $(t^{-}_{(i-1)6}, t^{-}_{i8})$ | crashing | ['crash', 'fix', 'give', 'keep', 'please', 'problem', 'star', 'think', 'time'] |
| $(t^{-}_{(i-1)7}, t^{-}_{i6})$ | login with Microsoft account | ['let', 'login', 'microsoft', 'time', 'try', 'turn', 'update', 'win'] |
| $(t^{-}_{(i-1)8}, t^{-}_{i6})$ | sign up with accounts | ['account', 'keep', 'make', 'sign'] |
| $(t^{-}_{(i-1)8}, t^{-}_{i9})$ | sync contacts and status update | ['add', 'contact', 'list', 'sync'] |

**Table 4.2** Similar Topics Identified among Reviews Before and After the Update

For Whatsapp, seven major updates are identified which divide the review dataset into seven subsets (i.e., $R_1$, $R_2$ ... $R_7$), each of which contains the reviews regarding one particular major update (i.e., $u_1$, $u_2$ ... $u_7$). Furthermore, each review subset is also divided based on sentiment analysis and NB classifier. Figure 4.5 shows an example of the identified similar topic chains.

Herein, positive topics are illustrated as blue circle while negative topic as red square. For example, $t^{+}_{1,7}$ is denoted as a blue circle marked "7" in column "T1" when $t^{-}_{2,10}$ is denoted as a red square marked "10" in column "T2".

The changes in the changes in users' opinions can be validated by the information from the update history of Whatsapp. For example, the "video call" feature added in Update 3 (Version 16.12.1), the "status" feature in update 4 (Version 17.3.14), and the "multiple contact cards" feature in update 5 (Version 17.4.18) all show that the developers noticed the constant negative opinions of users (i.e., the negative topic chain). Similarly, after Update 5, the complaints regarding contact and chat list can

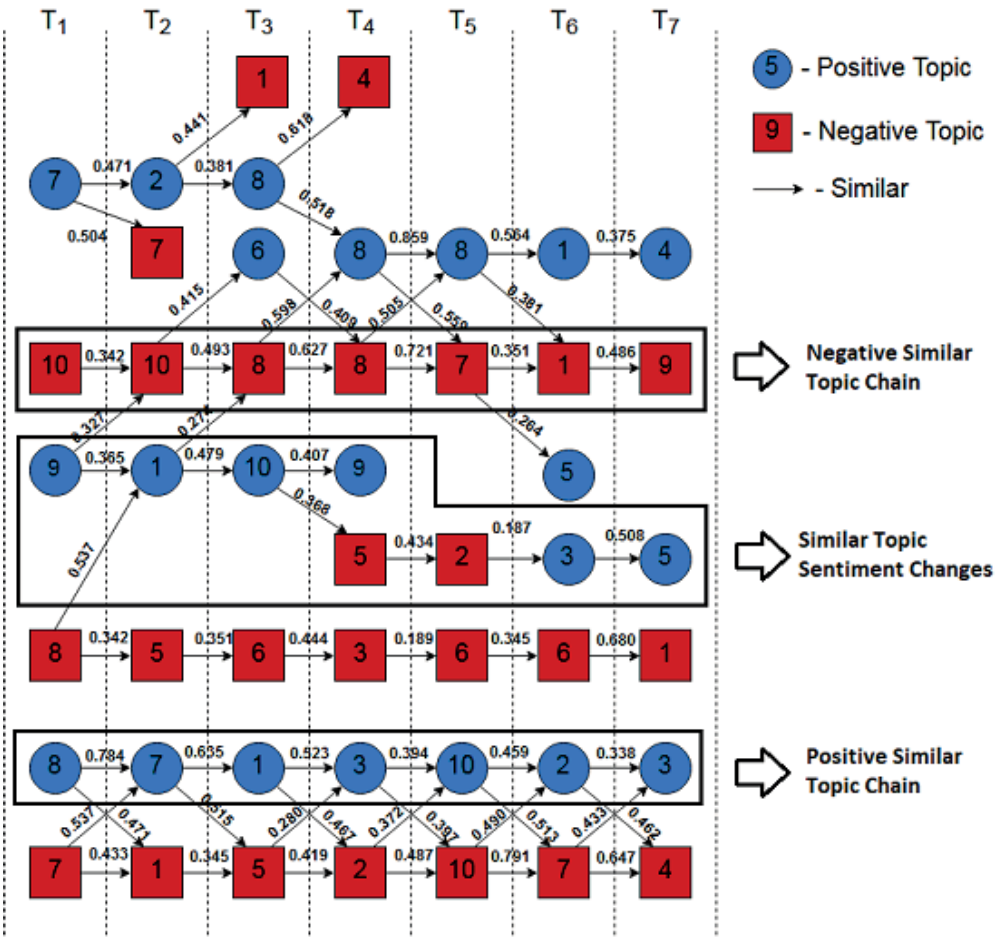


**Figure 4.5** Part of the Similar Topic Chains for Updates of Whatsapp

be observed until Update 6 where such complaint is not significant any more. The update history shows in Update 6 (Version 17.5.17) the developers added the "pin chat" feature to address the issue.

## 4.5 Context Analysis

The approaches of context analysis aims to facilitates the software evolution practices by providing additional useful information facilitating in-depth understanding of the environment, users and genres of the target software products. The context analysis approaches encompasses of three parts: 1) the analysis of situational contexts and

ways of interaction, 2) the analysis of user profiles towards preferences, and 3) the analysis of software types.

### 4.5.1 Situational Context and Ways of Interaction Analysis

The situational context refers to the extrinsic properties of the user and the app that impact the initiation of a user-app interaction. It encompasses of three aspects:

- **Temporal Context**: The user's sense of external time pressure of the user caused by the conflict or the accordance of his/her goal and the software's demands.
- **Spatial Context**: The user's current movement indicating the physical availability for the software usage.
- **Social Context**: The social norms that constrain the user from or encourage the user into the interaction with the software.

According to the three perspectives of situational context mentioned above, values are assigned to each perspective, combining which leads to a unique context scenario description (shown in Table 4.3). Based on the given situational context model, the situational context of user can be described by the combination of the three perspectives (i.e., 12 situational contexts).

| **Perspective** | **Value** |
|---|---|
| Temporal | **I**ntensive, **A**llocative |
| Spatial | **V**isiting, **T**raveling, **W**ondering |
| Social | **C**onstraining, **E**ncouraging |

**Table 4.3** Values of Situational Context Perspectives

On the other hand, the user-app interaction describes the various behavior patterns by which the users utilize a particular mobile app. Four ways of interaction, i.e., *intermittent*, *interrupting*, *accompanying*, and *ignoring* are proposed, based on the obtrusiveness and persistence nature of the features provided by the apps, as well as the way they fit in the process of the way of the user's original activity. By understanding the nature of the different situational contexts, a potentially proper mapping of each situational context with the ideal ways of interaction can be proposed.

The aim of situational contexts and ways of interaction analysis is to detect the potential conflicts between the ideal ways of interaction for the primary situational contexts of the target mobile app and the expected ways of interaction designed for the features of it. When such conflicts occur, it is likely hard for the users to enjoy the software perceived quality, which results in user churns. Therefore, to detect such conflicts, a series of steps are proposed, including: 1) Identify the primary situational contexts, 2) Specify the expected ways of interaction for each feature, 3) Compare the previous outcomes and identify the conflicts, and 4) Adjust the conflicting features towards mobility requirements.

### 4.5.2 User Type and Preference Analysis

The collective data of users' behaviors on the target software product or software products in general is useful asset to analyze the existing users' types and their preferences. Ideally, by acquiring such information shall enable the developers to better understand the user base and to update the software accordingly. Herein, exploratory factor analysis (EFA) is adopted within the proposed approach with the steps specified as follows.

**Step 1. Collect data.** The developers shall collect a statistically representative amount of data on the users' behaviors towards software features.

**Step 2. Verify sample adequacy.** Both Bartlett's Test of Sphericity [164] and Kaiser-Meyer-Olkin (KMO) Test [76] can be applied to verify the adequacy of the selected sample and furthermore the usefulness of applying factor analysis.

**Step 3. Determine the number of factors.** As a common pre-step of EFA, the number of factors shall be determined by parallel analysis (PA).

**Step 4. Detect and interpret factors.** With adequate samples of user behavior data, factors are detected and interpreted based on loadings that reflect the features belonging to each factor.

Publication IV applies the proposed method to a special case where users' online community behaviors are taken into account as features with the target to understand the different user types in terms of online video game community service features. Based on the result of EFA on the behavior data with 19 features of 60267 users, eight factors are determined and identified based on the factor loadings. Furthermore, based on the result of EFA, each of the eight factors can be interpreted

towards one of the unique preference attributes of the users in terms of online video game community service, shown in Table 4.4.

| Factor | Type | Related Features |
|---|---|---|
| 1 | Elite | Level, Badge, Friends, Profile Customization |
| 2 | Achiever | Games, Achievement, Perfect Games |
| 3 | Provider | Guides, Artworks |
| 4 | Completer | Showcases, Game Completion Rate |
| 5 | Improver | Workshop Items, Reviews |
| 6 | Trader | Item Owned, Trades Made, Market Transaction |
| 7 | Belonger | Groups, Profile Customization |
| 8 | Nostalgist | Screenshots, Videos |

**Table 4.4** Detected User Types and Related Features/Preferences

Such outcome demonstrates the validity of the approach using EFA to analyze the user groups of utilitarian software products provided such data is accessible.

### 4.5.3 Software Type Analysis

Although similarly the analysis of software types can also be conducted via EFA on software feature data, e.g., the game tags for video games [98], herein a social network analysis based approach is proposed with the aim to detect the hidden communities that connect the pre-defined software features and cluster such commonly processed features into most likely software types. By doing so, the developers shall have a clearer impression on the dominant types of software as well as what features each type encompasses of. The approach contains the following steps.

**Step 1. Collect and format data.** The first step is to collect all the software product data that contain explicit feature information. Therein, each single software can contain multiple features. To be noted, the terms used to describe features shall be well unified, so that no more than one feature is pointing to the same meaning.

**Step 2. Build network of software features.** When the original software feature data is ready, each connection between two particular software features are weighted by the number of software products that contain both features.

**Step 3. Detect common features.** The aim is to identify the software features

that are contained by nearly all the software products, e.g., icon, minimize, etc. The purpose is to obtain more effective community detection results in the later step. Herein, weighted degree, betweenness, closeness, and PageRank can all be used.

**Step 4. Detect communities.** After eliminating the common features, Louvain method is used to detect the best fitting communities where the modularity score is the highest.

Publication VI applies the approach to a special case where 23034 computer games data from Steam are collected with each attached with a set of tags that describe the features of the game. With such a dataset, a network with 326 vertices and 3035 edges is built. Therein, five tags, i.e., "Indie", "Action", "Adventure", "Singleplayer" and "Casual", rank the highest in all centrality measures and clearly commonly connect to majority of the other tags. Compared with the Top 15 tags by every metric, these five game tags have significantly higher value than the rest ten regardless.

These five tags are then removed from the network together with the edges on them and the 15 vertices are detected only connecting to these five tags. The Louvain method is applied on the remaining network with 306 vertices and 1813 edges with the highest modularity score of $Q = 0.414$ with four communities (shown in Figure 4.6) detected.

Therefore, based on the four communities detected, it is reasonable to verdict that computer games on Steam platform can be clustered into four types based on their features. By summarizing the tags/features with highest centrality (in this case, closeness is used as the measure), the four types of computer games can be summarized as follows, shown in Table 4.5.

To be noted, other centrality measures, such as, betweenness and PageRank, can also be used to determine the representing features of the types. In Publication VI, the results are to a large extent similar when different centrality measures are applied. Importantly, with such analysis on the software types together with the key features of each type, the developers shall have a better understanding and dynamic tracking of the existing software domain by latent categories.

To sum up, the three independent approaches of context analysis provide unique perspectives of understanding the contexts of software evolution, where developers can enrich their knowledge sources for decision making. To be noted, as marked in Figure 3.1, the methods for extracting formatted situational context data from user

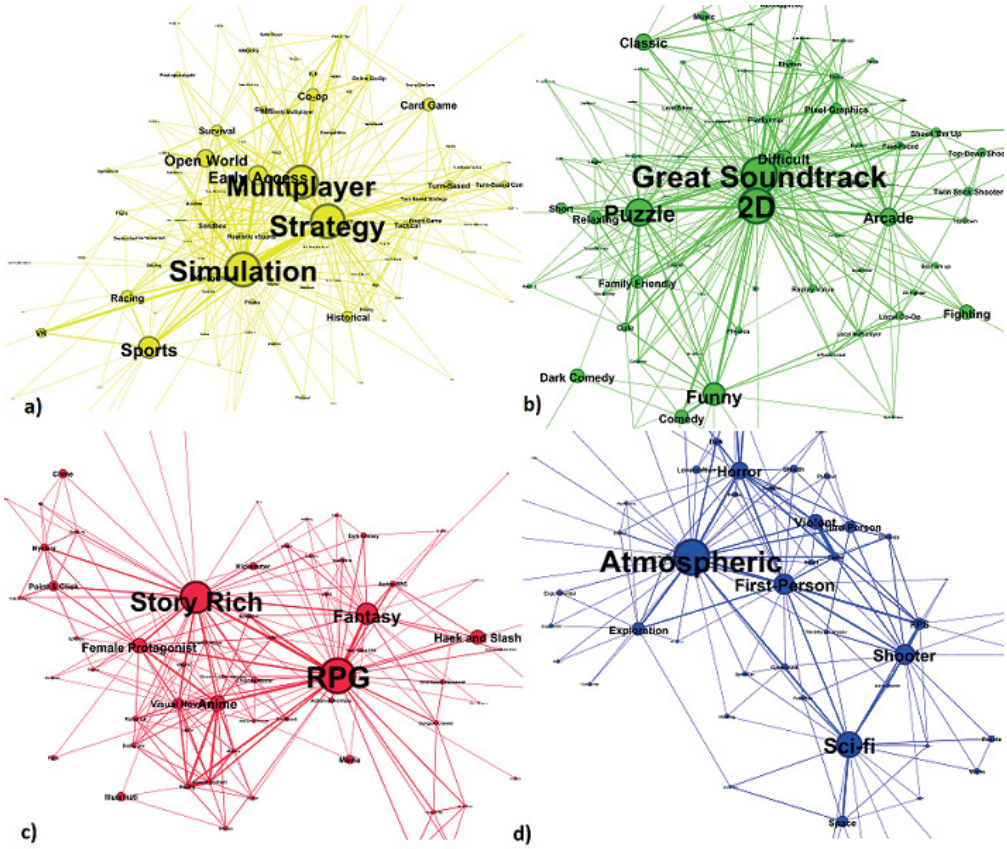


**Figure 4.6** The Detected Communities of Tags

reviews and the data-driven ways of analyzing situational contexts have not yet been explored. In addition, similar issue exists for the software features data for utilitarian software products instead of those for computer games, as well as software users behavior data instead of game players data. These shall be further discussed in the following chapter.

## 4.6 Summary of Publications

**Publication I** proposes a user-app interaction model for interpreting the situational context and ways of interaction for the use of mobile apps. The aim of the study is to provide a way of understanding the mobility of mobile apps in terms of the coordination of its primary situational contexts and the ideal ways of interaction.

| Strategy&Simulation | Puzzle&Arcade | RPG | Shooter |
|---|---|---|---|
| Strategy | 2D | RPG | Atmospheric |
| Simulation | Great Soundtrack | Story Rich | Sci-fi |
| Multiplayer | Puzzle | Fantasy | Exploration |
| Open World | Funny | Choices | First-Person |
| Early Access | Comedy | Text-based | Space |
| Survival | Relaxing | Dark Fantasy | Dystopian |
| Political | Classic | Kickstarter | Third Person |
| Medieval | Masterpiece | Action RPG | Hacking |
| Historical | Family Friendly | Mythology | Underwater |
| Realistic | Reply Value | Crime | Futuristic |

**Table 4.5** Computer Game Types Defined by High-Centrality Tags

The key of the proposed analysis approach is to detect the conflicts between the designed software features and the primary situational contexts where these features are commonly used. This paper performs as one perspective of the software evolution context analysis method toolkit. It facilitates such analysis towards specially the satisfaction of mobile app users by providing guidance in understanding their needs in terms of mobility, considering the mobility as the unique trait of mobile apps compared to software products in general.

**Publication II** proposes an analysis method to elicit user opinions regarding a particular software update by detecting the similar topics before and after this update. For any particular software update, regarding the collective users, their feedback shall change based on the update regardless of its positive or negative contribution to the overall quality of the software. The proposed approach herein focuses on four review subsets, i.e., the positive and negative reviews before and after the particular update. Thus, the changes of the users' collective needs and expectation shall be detected via the extracted topics of these review subsets from topic modeling and the similarity across these extracted topic sets. Such results shall provide intuitive knowledge on the consistent merits of the software, the unsolved issues, the uncomfortable changes and the improvement with this particular update. This paper contributes specifically to the software evolution monitoring approach by providing a key solution.

**Publication III**, as the extension for Publication II, aims to propose an approach of monitoring the software evolution timeline in terms of the changes of the detected users' needs and expectation through a series of software updates. The proposed approach shall enable the developers to identify the long-lasting issues of the mobile app by detecting the consistent user complaints, the newly emerging issues at a particular software update that results in users' collective negative feedback, and the resolved problems via the disappearance of collective negative user feedback. With this approach, users' needs and expectation are continuously monitored within the whole software evolution stage with potential issues being detected and addressed in time. The contribution of this paper is the core of the software evolution monitoring approach framework.

**Publication IV** provides a statistical analysis approach towards detecting the latent user types together with the preferred behaviors of each type of users. The approach adopts the exploratory factor analysis on the large dataset of the users' behaviors on a particular software product or service. Though the case in this paper is focusing on the understanding of the community behaviors of computer game players, the contribution can be extensively seen as the analysis of the online user behaviors towards online community services. Therefore, by understanding the potential user types and the according preferences and behaviors, the developers can therefore take action during software evolution targeting the popular or profitable user behaviors towards enhanced perceived quality and user satisfaction. This paper also performs as one key perspective of the software evolution context analysis method toolkit.

**Publication V** aims to provide a statistical approach to identify the problematic software update by detecting the abnormal sentiment changes, i.e., a sudden increase of negative opinions, in the collective user reviews. The approach adapts the statistical anomaly detection of distribution where abnormal update is identified by the similarity of the top frequent keywords of reviews on abnormal day and the ones of one of its former updates. This approach facilitates the developers in terms of software evolution monitoring by provide an effective solution to identify the significantly problematic updates that require attention. Such a problematic update, if not being detected quickly and tackled properly, can cause damaging negativity in users' opinions and increasing user churn.

**Publication VI** proposes an approach of using social network analysis techniques, e.g., centrality analysis and community detection, to analyze the latent software types based on software features. Though in the publication, the approach is demonstrated by the application on computer game genre analysis based on game tags data, the approach itself can be applied extensively with utilitarian software data. With the contribution of this paper, the developers shall be able to obtain the overview of the existing software product market by various software types. It can also facilitate them by providing a horizontal comparison between their product and the ones in the same category for potential enlightenment. This paper also performs as one key perspective of the software evolution context analysis method toolkit.

**Publication VII** provides an approach of the evaluating, quantifying and visualizing the perceived quality of a particular software product based on the sentiment analysis of its end user reviews. It also provides a data-driven solution to detect the specific merits and defects of the software in terms of a pre-defined quality framework using Bayes classification and LDA topic modeling. Though the publication verifies the proposed approach via the case study of analyzing the playability (perceived quality of games) of a particular computer game (hedonic software product for entertainment purpose), the approach shall be conduct effectively provided having the user review data on utilitarian software. The contribution of this paper serves as the core of this research by providing key solutions to both perceived quality analysis and collective user needs and expectation analysis. By combining this approach to the ones proposed in Publication II and III, the approaches can be synthesized towards a review data driven approach to continuous monitor the software evolution in terms of the confirmation of user expectation and needs by the perceived software quality.

# 5 DISCUSSION

In this chapter, the results of the research are discussed where each previously proposed research question shall be accordingly answered. Further in Section 5.2, the contribution of the research is summarized. In Section 5.3, the discussion shall cover the limitation and future works.

## 5.1 Answering Research Questions

**RQ1. How to analyze users' collective expectation and perceived quality in use with data-driven approaches by exploiting sentiment and topics?**

Towards answering RQ1, this research focuses on using sentiment analysis and topic modeling techniques to evaluate the perceived quality of a software product and to extract the users' collective expectations and needs to it based on a large volume of user reviews. The purpose of this approach is to effectively support the meaningful maintenance and update during the software evolution stage by understanding the end users' needs and sustaining their satisfaction, as, knowingly, the user satisfaction is formed based on the expectation and the confirmation of such expectation towards the users' perception of performance.

To evaluate the overall perceived quality of a software product based on user reviews, the proposed approach first identifies the informative reviews using a pre-trained Bayes classifier after preprocessing the raw reviews. The second step is to classify the obtained informative reviews into pre-defined aspects based on a selected quality framework using multi-label text classification techniques. Then, for each quality aspect, the average sentiment is calculated based on the classified reviews which reflect the perceived quality of the aspects. Finally, the perceived quality of the product can be visualized into a radar diagram with the quantified evaluation for each aspect visible.

On the other hand, the collective expectation and needs of the end-users can be extracted via topic modeling, which is also proposed in the approach. With the previously filtered and classified informative reviews, positive and negative reviews are separated into subsets using sentiment analysis. For each review subset, topics are then extracted that reflect the collective needs and complaints in terms of each quality aspect from the end-users.

**RQ2. How to monitor user satisfaction over software updates during software evolution using reviews' topics and sentiments?**

To answer RQ2, this research presents a two-fold data-driven approach 1) to observe the trends and identify the changes in the collective feedback of the users, and 2) to detect the problematic update which severely evokes the users' negativity. The approach can use the output of the method for RQ1 as input, that is, the positive and negative topic sets extracted at the selected time periods, e.g., between major updates, and the series of quantified perceived quality through software evolution. The key mechanism for the proposed approach is, according to the ECT, to verify the confirmation of the detected user expectations via the comparison between the topics before and after the updates. For any two adjacent time periods divided by major updates, the latent hypothesis is that the changes in users' opinions between the two review sets reflect the perceived quality of the particular update and that particular version of the software. Therefore, ideally, by tracking such changes through the timelines of the updates in the software evolution stage, the developers shall be able to monitor the users' satisfaction over the software with the specific changes in their needs.

Specifically, to monitor user satisfaction during the software evolution stage, the proposed approach indicates the users' constant positive opinions on particular features (i.e., topics) can be reflected by the positive similar topic chains through the updates. On the contrary, the unsolved issues can then be reflected by the negative ones. In addition, the similar topics between positive and negative topics shall reflect the changes of users' opinions on the specific feature, indicating either a positive enhancement or a fail attempt. On the other hand, the approach also enables to detect the abnormal decreases in quantified perceived quality through software evolution timeline using statistical analysis, using statistical anomaly detection of distribution. Furthermore, the identified abnormal days are then mapped to a causing former up-

date by comparing the similarity between the top frequent keywords of the reviews on that day and the description of the update.

**RQ3. How to analyze users' profiles, software types and situational contexts as the contexts of use that support the above activities?**

The answer to RQ3 is three-fold, including solutions to the following perspectives: 1) situational contexts and ways of interaction analysis towards mobility conflict detection, 2) user types and preference analysis based on software behavior data, and 3) software type and related features analysis based on software feature data.

To analyze the situational contexts of software products, especially mobile apps, the research firstly introduces the concept of ways of interaction. The hypothesis lies where for each specific software feature the users shall interact with it in one or multiple ideal ways in order to achieve their satisfaction. Therefore, the approach aims to introduce the notion that such conflicts in the expected ways of interaction for the primary software situational contexts and the ideal ways of interaction for software features should be identified and addressed.

On the other hand, the user types and preference analysis approach focuses on understanding the latent software user types and the according preferences of each type of user. Given an adequate amount of user behavior data, using EFA, the approach shall enable the developers to gain such context information of their users and insights on how to conduct proper maintenance strategy accordingly.

Furthermore, the software type analysis approach uses centrality measure and community detection from social network analysis to detect the latent software types within the market with the data of software feature data. The approach shall facilitate the understanding of the software ecology; when knowing the type of a particular software product, the developers shall easily gain insights on the commonly adopted features within the category and the potential features to be added.

## 5.2 Research Contribution

The contribution of this research towards answering the research questions regarding mending the research gaps.

| RQ | Research Gap | Contribution |
|---|---|---|
| RQ1 | *Difficulty in involving end users and extracting their feedback* | This research contributes to provide a data-driven approach to extract the users' collective opinions, i.e., their needs and expectations, on large volume of user reviews. In addition, the approach also provides quantified evaluation of users' collective perceived quality of the software using sentiment analysis. In this way, effective user involvement and opinion extraction can be achieved. Furthermore, a statistical approach on identifying the abnormal update is also proposed facilitating specially on the extraction of users' urgent needs. This contribution is published in Publication VII and V. |
| RQ2 | *Lack of methods in continuous monitoring evolution status via the changes in users' collective needs* | This research also contributes to the monitoring of software evolution status via tracking the status of users' collective expectation throughout software releases. Based on the detection of similar topic pairs in different sentiment group before and after each release, the changes within software evolution process can be detected in terms of the changes in users' collective satisfaction. This contribution is published in Publication II and III. |
| RQ3 | *Lack of methods in analyzing various context information* | This research also contributes to the data-driven analysis of important context of use information of software products, which supports the analysis of users' expectation and satisfaction. The context information herein include the situational contexts of software use, user profiles and preferences, and software types. This contribution is published in Publication I, IV and VI. |

**Table 5.1** Summary of Research Contribution

Furthermore, the implication of this research, in terms of both academia and industry, can be seen as follows.

### Contribution to the Academia

From the academic perspective, this thesis work contributes to the examination of applying data-driven end-user review analysis methods supporting software maintenance and evolution. The main implication is to enrich the existing domain knowledge of software maintenance and evolution in terms of taking advantage of the collective intelligence of the end-users. Though, as mentioned in the background and related work, many studies have proposed data-driven approaches using similar techniques using user reviews, seldom do they focus on solving specific problems regarding software maintenance and evolution, especially the continuous nature of it. In addition, the thesis conveys a unique contribution to the research on software evolution contexts, including 1) the situational contexts, 2) user types and preferences, and 3) software types and related features. These shall not only enrich the existing domain knowledge of software maintenance and evolution largely but also can lead to potential interdisciplinary contribution with other domains.

### Contribution to the Industry

This thesis also contributes to the software maintenance and evolution practice even in the larger scope of the software industry by proposing an effective series of approaches that address the critical issues within. Currently, though software maintenance and evolution is recognized as important to the sustainability of nearly any software product, the practical solutions to help the developers in terms of effectiveness and continuity are limited. Together with the rapidly growing maturity in data mining, machine learning, and NLP techniques, effectively and continuously understanding the needs and expectations of a large number of end users is thus possible. In addition, the thesis also provides approaches to understand the latent context information of software evolution, i.e., situations, user types and preferences, software types, and related features. The main benefit of this practice is that developers can easily obtain additional information guiding their maintenance strategy, release planning, and other related decision-making.

## 5.3 Limitation & Future Work

Despite the contribution of the thesis, it does fall short in various aspects which can be enhanced in future studies. Firstly, the application of the proposed approaches to a utilitarian software product is yet to be verified, as the case studies conducted in Publication IV, VI, and VII use the data of computer games and the game players. Though none of the approaches adopts the hedonic characteristics of video games as influential factors, it still can be more convincing provided such cases use the data across genres. Meanwhile, to be emphasized, the reason for the above-mentioned choices is due to the lack of accessibility to a large volume of software user review data.

Secondly, as shown in Figure 3.1, neither has the approach of classifying informative situational contexts from user reviews yet been studied, nor the sources or ways of crawling situational contexts data is provided. One of the critical reasons is that it is not a common practice for the end-users to report their context information in the reviews. Due to the necessity of user privacy protection, collecting user contexts is to a certain extent illegal or unethical. In addition, it is also difficult to collect such data without affecting the users' experience. Though such information can be collected by particular companies, due to the above reasons such information cannot be shared publicly. More importantly, the connection and application of the proposed context analysis approaches to the software evolution monitoring is yet to be explored. Though the approach enables to detect the latent context information on user types and preferences as well as software types and related features, it has not yet been suggested how to categorize a particular software product to a certain type nor how to identify a certain type of users from a large user pool. These aspects are important to the practicality of the research and shall be conducted in future work.

Thirdly, the proposed approach uses update time as the separator of reviews to subsets, which might evoke issues. It is likely the users can send reviews after a particular release when still using the old version. However, though the precise percentage of such incidents cannot be estimated, the effectiveness of the proposed approach shall not deteriorate. Nonetheless, it is still recommended that an additional mechanism is added to label the target software version when the users are giving reviews. On the other hand, besides the limitation mentioned above, the following future work shall be conducted to enrich the according domain knowledge and to expand

the implication of this research.

First of all, the direct supplement to the existing framework is to investigate the ways of utilizing the context information in the monitoring and planning for software evolution. Ideally, the users' profile and behavior data shall be mapped to the data of their preferences to software types and the relevant features, which is similar to the synthesis of the review-based user profile building and review-based product profile building [23]. In this way, the reviews, which are only classified by the informativeness currently, can be further weighted based on the key relevant behaviors of the reviewers and the importance of the software features being targeted, which shall lead to more audience-sensitive outcomes. More specifically, to achieve such results, identifying the type of each individual end-user and the type of the target software product requires further investigation as well. For the user profile data, a potentially practical data-driven way towards such a solution is to use principal component analysis [188] to reduce the data dimension according to the user types and assign each user into the closest type.

Secondly, serve as part of the trend towards the application of AI in software engineering, this research shall be continued towards the broader contribution therein. With the benefits in automation, using AI-based techniques shall have a positive influence on the software development, testing, and other decision-making activities [128]. Together with AI techniques, the collective intelligence of the crowd is the source of information that can be used to support the key software development activities as well [196].

Furthermore, the contribution of this study can also explore the potential boundary of interdisciplinary studies towards the application of computational methods on game studies in the near future. The unique advantage of data science research methods, compared to the traditional, is the capability of extracting latent knowledge from the existing phenomena via mining from statistically massive data. Such findings shall largely enrich a particular knowledge domain with either the verification of existing knowledge or the detection of the new. Taking the trending domain of computational social science [27] as an example, the combination of data science and traditional social science has contributed greatly to the detecting of many social patterns and issues, as well as the novel understanding of social phenomena. As a multidisciplinary field of study and learning with games and related phenomena as its subject matter, game studies can certainly gain support from such novel data-

driven methods towards new findings in the study of games, the study of players, and the study of the contexts of the previous two [118]. Moreover, millions of players on game platforms, e.g., Steam, have been continuously producing a massive volume of data measured in TB, which can be conveniently accessed via API. These data could be invaluable sources supporting such research.

# 6 CONCLUSION

Software maintenance and evolution is a critical and long-lasting stage of any commercial software product, where, due to the contemporary competitive market, end users' needs and expectations must be carefully monitored and handled. Maintaining the satisfaction of the software user base is one of the keys to the long-term success of any software product. To such an end, accompanied with the rapid growth in user review data and the advance in data science technologies, such computational methods with a large volume of data provides effective solutions to evaluate users' satisfaction and detect their needs. Therefore, the core contribution of this thesis is to explore the data-driven ways of continuously assessing users' satisfaction, extracting their needs, and monitoring the status of changes therein during the software evolution process.

Specifically, the thesis contributes to three aspects of the main research theme, that is, to apply a data-driven approach on continuous monitoring of the software evolution status via exploring the changes in end-user satisfaction and needs. Firstly, the research tackles the analysis of users' collective expectation and perceived quality in use with data-driven approaches by exploiting sentiment and topics. With sentiment analysis and Bayes classification on identified informative user reviews, the overall perceived quality of the users can be quantified and visualized in terms of a pre-defined framework. Furthermore, the needs and complaints about the users can be extracted via topic modeling on each of the aspects. Secondly, the research provides an approach of assessing the changes in the collective users' perceived quality and needs via the identification of similar topic pairs in positive and negative review sets before and after software updates and that of distribution anomaly in review sentiment. Via the application of the approach through the multiple updates in the evolution stage, the changes of opinions in the user base can be efficiently monitored with even abnormally negative changes detected in time. In addition, the third aspect of the thesis contribution explores the potential context information that can

be utilized to support the above activities. The context information includes software situational contexts and ways of interaction, user profiles and preferences on software features, and software types and related features. Though the practical impact of applying such context information is yet to be further investigated, weighted evaluation with the personalized solution can be expected.

# REFERENCES


[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith and P. Steggles. Towards a better understanding of context and context-awareness. *International symposium on handheld and ubiquitous computing*. Springer. 1999, 304–307.

[2] D. Alonso-Rıéos, A. Vázquez-Garcıéa, E. Mosqueira-Rey and V. Moret-Bonillo. A context-of-use taxonomy for usability studies. *International Journal of Human-Computer Interaction* 26.10 (2010), 941–970.

[3] E. W. Anderson and M. W. Sullivan. The antecedents and consequences of customer satisfaction for firms. *Marketing science* 12.2 (1993), 125–143.

[4] M. Aoyama. Continuous and discontinuous software evolution: aspects of software evolution across multiple product lines. *Proceedings of the 4th international workshop on Principles of software evolution*. 2001, 87–90.

[5] D. Baccarini, G. Salm and P. E. Love. Management of risks in information technology projects. *Industrial Management & Data Systems* (2004).

[6] D. J. Bartholomew, M. Knott and I. Moustaki. *Latent variable models and factor analysis: A unified approach*. Vol. 904. John Wiley & Sons, 2011.

[7] K. H. Bennett and V. T. Rajlich. Software maintenance and evolution: a roadmap. *Proceedings of the Conference on the Future of Software Engineering*. 2000, 73–87.

[8] N. Bevan. Quality in use: Meeting user needs for quality. *Journal of systems and software* 49.1 (1999), 89–96.

[9] N. Bevan and M. Macleod. Usability measurement in context. *Behaviour & information technology* 13.1-2 (1994), 132–145.

[10] A. Bhattacherjee. Understanding information systems continuance: An expectation-confirmation model. *MIS quarterly* (2001), 351–370.

[11] D. M. Blei, A. Y. Ng and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research* 3 (2003), 993–1022.

[12] V. D. Blondel, J.-L. Guillaume, R. Lambiotte and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008.10 (2008), P10008.

[13] B. W. Boehm. A spiral model of software development and enhancement. *Computer* 21.5 (1988), 61–72.

[14] B. W. Boehm, J. R. Brown and M. Lipow. Quantitative evaluation of software quality. *Proceedings of the 2nd international conference on Software engineering*. 1976, 592–605.

[15] U. Brandes, S. P. Borgatti and L. C. Freeman. Maintaining the duality of closeness and betweenness centrality. *Social Networks* 44 (2016), 153–159.

[16] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems* 30.1-7 (1998), 107–117.

[17] M. Burnett, C. Cook and G. Rothermel. End-user software engineering. *Communications of the ACM* 47.9 (2004), 53–58.

[18] H. Cai, X. Fu and A. Hamou-Lhadj. A study of run-time behavioral evolution of benign versus malicious apps in android. *Information and Software Technology* 122 (2020), 106291.

[19] R. N. Cardozo. An experimental study of customer effort, expectation, and satisfaction. *Journal of marketing research* 2.3 (1965), 244–249.

[20] L. V. G. Carreno and K. Winbladh. Analysis of user comments: an approach for software requirements evolution. *2013 35th international conference on software engineering (ICSE)*. IEEE. 2013, 582–591.

[21] R. B. Cattell. The meaning and strategic use of factor analysis. *Handbook of multivariate experimental psychology*. Springer, 1988, 131–203.

[22] P. L. Chebyshev. Des valeurs moyennes, Liouville's. *J. Math. Pures Appl.* 12 (1867), 177–184.

[23] L. Chen, G. Chen and F. Wang. Recommender systems based on user reviews: the state of the art. *User Modeling and User-Adapted Interaction* 25.2 (2015), 99–154.

[24] M. Chen and X. Liu. Predicting popularity of online distributed applications: iTunes app store case analysis. *Proceedings of the 2011 iConference*. 2011, 661–663.

[25] N. Chen, J. Lin, S. C. Hoi, X. Xiao and B. Zhang. AR-miner: mining informative reviews for developers from mobile app marketplace. *Proceedings of the 36th international conference on software engineering*. 2014, 767–778.

[26] C. M. Cheung, M. K. Lee and N. Rabjohn. The impact of electronic word-of-mouth: The adoption of online opinions in online customer communities. *Internet research* (2008).

[27] C. Cioffi-Revilla. Introduction to computational social science. *London and Heidelberg: Springer* (2014).

[28] A. Cockburn and J. Highsmith. Agile software development, the people factor. *Computer* 34.11 (2001), 131–133.

[29] J. B. Cohen and M. E. Goldberg. The dissonance model in post-decision product evaluation. *Journal of Marketing Research* 7.3 (1970), 315–321.

[30] C. K. Coursaris and D. J. Kim. A meta-analytical review of empirical mobile usability studies. *Journal of usability studies* 6.3 (2011), 117–171.

[31] R. Darimont and A. Van Lamsweerde. Formal refinement patterns for goal-driven requirements elaboration. *ACM SIGSOFT Software Engineering Notes* 21.6 (1996), 179–190.

[32] P. De Meo, E. Ferrara, G. Fiumara and A. Provetti. Generalized louvain method for community detection in large networks. *2011 11th International Conference on Intelligent Systems Design and Applications*. IEEE. 2011, 88–93.

[33] P. Debois et al. Devops: A software revolution in the making. *Journal of Information Technology Management* 24.8 (2011), 3–39.

[34] J. DeCoster. Overview of factor analysis. (1998).

[35] H. Desurvire, M. Caplan and J. A. Toth. Using heuristics to evaluate the playability of games. *CHI'04 extended abstracts on Human factors in computing systems*. 2004, 1509–1512.

[36] A. Di Sorbo, G. Grano, C. Aaron Visaggio and S. Panichella. Investigating the criticality of user-reported issues through their relations with app rating. *Journal of Software: Evolution and Process* 33.3 (2021), e2316.

[37] A. Di Sorbo, S. Panichella, C. V. Alexandru, J. Shimagaki, C. A. Visaggio, G. Canfora and H. 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*. 2016, 499–510.

[38] W. Dzida, S. Herda and W. D. Itzfeldt. User-perceived quality of interactive systems. *IEEE Transactions on Software Engineering* 4 (1978), 270–276.

[39] S. Erevelles, S. Srinivasan and S. Rangel. Consumer satisfaction for internet service providers: an analysis of underlying processes. *Information Technology and Management* 4.1 (2003), 69–89.

[40] M. Ester, H.-P. Kriegel, J. Sander, X. Xu et al. A density-based algorithm for discovering clusters in large spatial databases with noise. *Kdd*. Vol. 96. 34. 1996, 226–231.

[41] A. Finkelstein, M. Harman, Y. Jia, W. Martin, F. Sarro and Y. Zhang. Investigating the relationship between price, rating, and popularity in the Blackberry World App Store. *Information and Software Technology* 87 (2017), 119–139.

[42] A. Finkelstein, M. Harman, Y. Jia, F. Sarro and Y. Zhang. Mining app stores: Extracting technical, business and customer rating information for analysis and prediction. *RN* 13 (2013), 21.

[43] B. Fitzgerald and K.-J. Stol. Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software* 123 (2017), 176–189.

[44] L. C. Freeman. Centrality in social networks conceptual clarification. *Social networks* 1.3 (1978), 215–239.

[45] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong and N. Sadeh. Why people hate your app: Making sense of user feedback in a mobile app store. *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2013, 1276–1284.

[46] N. Genc-Nayebi and A. Abran. A systematic literature review: Opinion mining studies from mobile app store user reviews. *Journal of Systems and Software* 125 (2017), 207–219.

[47] A. Ghose and P. G. Ipeirotis. Estimating the helpfulness and economic impact of product reviews: Mining text and reviewer characteristics. *IEEE transactions on knowledge and data engineering* 23.10 (2010), 1498–1512.

[48] H. L. Gibson. Determining user involvement. *Journal of Systems Management* 28.8 (1977), 20–22.

[49] M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences* 99.12 (2002), 7821–7826.

[50] M. W. Godfrey and D. M. German. The past, present, and future of software evolution. *2008 Frontiers of Software Maintenance*. IEEE. 2008, 129–138.

[51] M. W. Godfrey and D. M. German. On the evolution of Lehman's Laws. *Journal of Software: Evolution and Process* 26.7 (2014), 613–619.

[52] J. B. Gotlieb, D. Grewal and S. W. Brown. Consumer satisfaction and perceived quality: complementary or divergent constructs?: *Journal of applied psychology* 79.6 (1994), 875.

[53] E. C. Groen, J. Doerr and S. Adam. Towards crowd-based requirements engineering a research preview. *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer. 2015, 247–253.

[54] E. C. Groen, N. Seyff, R. Ali, F. Dalpiaz, J. Doerr, E. Guzman, M. Hosseini, J. Marco, M. Oriol, A. Perini et al. The crowd in requirements engineering: The landscape and challenges. *IEEE software* 34.2 (2017), 44–52.

[55] J. Grudin. Interactive systems: Bridging the gaps between developers and users. *Computer* 24.4 (1991), 59–69.

[56] J. Grudin. Systematic sources of suboptimal interface design in large product development organizations. *Human-computer interaction* 6.2 (1991), 147–196.

[57] E. Guzman and W. Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. *2014 IEEE 22nd international requirements engineering conference (RE)*. IEEE. 2014, 153–162.

[58] J. F. Hair, W. C. Black, B. J. Babin, R. E. Anderson, R. L. Tatham et al. *Multivariate data analysis (Vol. 6)*. 2006.

[59] M. Harman, Y. Jia and Y. Zhang. App store mining and analysis: MSR for app stores. *2012 9th IEEE working conference on mining software repositories (MSR)*. IEEE. 2012, 108–111.

[60] G. Hecht, O. Benomar, R. Rouvoy, N. Moha and L. Duchien. Tracking the software quality of android applications along their evolution (t). *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE. 2015, 236–247.

[61] J. Heiskari and L. Lehtola. Investigating the state of user involvement in practice. *2009 16th Asia-Pacific Software Engineering Conference*. IEEE. 2009, 433–440.

[62] E. A. Holbrook, J. H. Hayes and A. Dekhtyar. Toward automating requirements satisfaction assessment. *2009 17th IEEE International Requirements Engineering Conference*. IEEE. 2009, 149–158.

[63] J. L. Horn. A rationale and test for the number of factors in factor analysis. *Psychometrika* 30.2 (1965), 179–185.

[64] K. Hotta, Y. Sano, Y. Higo and S. Kusumoto. Is duplicate code more frequently modified than non-duplicate code in software evolution? An empirical study on open source software. *Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE)*. 2010, 73–82.

[65] C. Hutto and E. Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. *Proceedings of the International AAAI Conference on Web and Social Media*. Vol. 8. 1. 2014.

[66] C. Iacob, V. Veerappa and R. Harrison. What are you complaining about?: a study of online reviews of mobile applications. *27th International BCS Human Computer Interaction Conference (HCI 2013) 27*. 2013, 1–6.

[67] ISO/IEC. *IEEE International Standard for Software Engineering - Software Life Cycle Processes - Maintenance*. Standard 14764:2006. International Organization for Standardization, 2006.

[68] ISO/IEC. *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. Standard 25010:2011. International Organization for Standardization, 2011.

[69] ISO/IEC. *Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts*. Standard 9241-11:2018. International Organization for Standardization, 2018.

[70] ISO/IEC. *Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems*. Standard 9241-210:2019. International Organization for Standardization, 2019.

[71] ISO/IEC/IEEE. *ISO/IEC/IEEE International Standard - Systems and software engineering–Vocabulary*. Standard 24765-2017. IEEE Standards Association, 2017.

[72] N. Jindal and B. Liu. Opinion spam and analysis. *Proceedings of the 2008 international conference on web search and data mining*. 2008, 219–230.

[73] M. Jo. *Google-Play-Scraper*. `https://github.com/JoMingyu/google-play-scraper`. 2021.

[74] S. Jumisko-Pyykkö and T. Vainio. Framing the context of use for mobile HCI. *International journal of mobile human computer interaction (IJMHCI)* 2.4 (2010), 1–28.

[75] H. F. Kaiser. The varimax criterion for analytic rotation in factor analysis. *Psychometrika* 23.3 (1958), 187–200.

[76] H. F. Kaiser. An index of factorial simplicity. *Psychometrika* 39.1 (1974), 31–36.

[77] H. Khalid. On identifying user complaints of iOS apps. *2013 35th international conference on software engineering (ICSE)*. IEEE. 2013, 1474–1476.

[78] H. Khalid, E. Shihab, M. Nagappan and A. E. Hassan. What do mobile app users complain about?: *IEEE software* 32.3 (2014), 70–77.

[79] M. Khurum, K. Petersen and T. Gorschek. Extending value stream mapping through waste definition beyond customer perspective. *Journal of Software: Evolution and Process* 26.12 (2014), 1074–1105.

[80] H.-W. Kim, H.-L. Lee and S.-J. Choi. An Exploratory Study on the Determinants of Mobile Application Purchase. *The Journal of Society for e-Business Studies* 16 (Nov. 2011). DOI: `10.7838/jsebs.2011.16.4.173`.

[81] S.-M. Kim, P. Pantel, T. Chklovski and M. Pennacchiotti. Automatically assessing review helpfulness. *Proceedings of the 2006 Conference on empirical methods in natural language processing*. 2006, 423–430.

[82] A. Knauss. On the usage of context for requirements elicitation: End-user involvement in IT ecosystems. *2012 20th IEEE International Requirements Engineering Conference (RE)*. IEEE. 2012, 345–348.

[83] A. J. Ko, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, C. Scaffidi, J. Lawrance, H. Lieberman, B. Myers et al. The state of the art in end-user software engineering. *ACM Computing Surveys (CSUR)* 43.3 (2011), 21.

[84] A. J. Ko, M. J. Lee, V. Ferrari, S. Ip and C. Tran. A case study of post-deployment user feedback triage. *proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering*. 2011, 1–8.

[85] N. Korfiatis, E. Garcıéa-Bariocanal and S. Sánchez-Alonso. Evaluating content quality and helpfulness of online product reviews: The interplay of review helpfulness vs. review content. *Electronic Commerce Research and Applications* 11.3 (2012), 205–217.

[86] J. Koskinen, H. Lahtonen and T. Tilus. Software maintenance cost estimation and modernization support. *ELTIS-project, University of Jyväskylä* (2003).

[87] S. Kujala. User involvement: a review of the benefits and challenges. *Behaviour & information technology* 22.1 (2003), 1–16.

[88] S. Kujala, M. Kauppinen, L. Lehtola and T. Kojo. The role of user involvement in requirements quality and project success. *13th IEEE International Conference on Requirements Engineering (RE'05)*. IEEE. 2005, 75–84.

[89] B. V. Kumar and L. Hassebrook. Performance measures for correlation filters. *Applied optics* 29.20 (1990), 2997–3006.

[90] M. M. Lehman. Laws of software evolution revisited. *European Workshop on Software Process Technology*. Springer. 1996, 108–124.

[91] M. M. Lehman. Programs, cities, students—limits to growth?: *Programming Methodology*. Springer, 1978, 42–69.

[92] M. M. Lehman. On understanding laws, evolution, and conservation in the large-program life cycle. *Journal of Systems and Software* 1 (1979), 213–221.

[93] M. M. Lehman. Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE* 68.9 (1980), 1060–1076.

[94] M. M. Lehman. Feedback in the software evolution process. *Information and Software technology* 38.11 (1996), 681–686.

[95] M. Lehman. Software's future: Managing evolution. *IEEE software* 15.1 (1998), 40.

[96] L. Lessig, C. Shirky, M. Cusumano et al. *Perspectives on free and open source software*. MIT press, 2005.

[97] E. Letier and A. Van Lamsweerde. Reasoning about partial goal satisfaction for requirements and design engineering. *Proceedings of the 12th ACM SIGSOFT twelfth international symposium on Foundations of software engineering*. 2004, 53–62.

[98] X. Li. Towards Factor-oriented Understanding of Video Game Genres using Exploratory Factor Analysis on Steam Game Tags. *2020 IEEE International Conference on Progress in Informatics and Computing (PIC)*. IEEE. 2020, 207–213.

[99] X. Li, C. Lu, J. Peltonen and Z. Zhang. A statistical analysis of Steam user profiles towards personalized gamification. *International GamiFIN Conference (GamiFIN)*. CEUR-WS. 2019.

[100] X. Li and B. Zhang. A preliminary network analysis on steam game tags: another way of understanding game genres. *Proceedings of the 23rd International Conference on Academic Mindtrek*. 2020, 65–73.

[101] X. Li, B. Zhang, Z. Zhang and K. Stefanidis. A Sentiment-Statistical Approach for Identifying Problematic Mobile App Updates Based on User Reviews. *Information* 11.3 (2020), 152.

[102] X. Li and Z. Zhang. A User-App Interaction Reference Model for Mobility Requirements Analysis. *International Conference on Software Engineering Advances (ICSEA)*. 2015, 170–177.

[103] X. Li, Z. Zhang and J. Nummenmaa. Models for mobile application maintenance based on update history. *2014 9th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*. IEEE. 2014, 1–6.

[104] X. Li, Z. Zhang and K. Stefanidis. Mobile App Evolution Analysis Based on User Reviews. *International Conference on Intelligent Software Methodologies, Tools, and Techniques (SoMeT)*. 2018, 773–786.

[105] X. Li, Z. Zhang and K. Stefanidis. Sentiment-aware Analysis of Mobile Apps User Reviews Regarding Particular Updates. *International Conference on Software Engineering Advances (ICSEA)*. 2018, 109.

[106] X. Li, Z. Zhang and K. Stefanidis. A Data-Driven Approach for Video Game Playability Analysis Based on Players' Reviews. *Information* 12.3 (2021), 129.

[107] T.-P. Liang, H.-J. Lai and Y.-C. Ku. Personalized content recommendation and user satisfaction: Theoretical synthesis and empirical findings. *Journal of Management Information Systems* 23.3 (2006), 45–70.

[108] T.-P. Liang and Y.-H. Yeh. Effect of use contexts on the continuous use of mobile services: the case of mobile games. *Personal and Ubiquitous Computing* 15.2 (2011), 187–196.

[109] B. P. Lientz, E. B. Swanson and G. E. Tompkins. Characteristics of application software maintenance. *Communications of the ACM* 21.6 (1978), 466–471.

[110] B. P. Lientz and E. B. Swanson. *Software maintenance management*. Addison-Wesley Longman Publishing Co., Inc., 1980.

[111] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu and H. W. Lauw. Detecting product review spammers using rating behaviors. *Proceedings of the 19th ACM international conference on Information and knowledge management*. 2010, 939–948.

[112] G. Lindgaard, R. Dillon, P. Trbovich, R. White, G. Fernandes, S. Lundahl and A. Pinnamaneni. User Needs Analysis and requirements engineering: Theory and practice. *Interacting with computers* 18.1 (2006), 47–70.

[113] B. Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies* 5.1 (2012), 1–167.

[114] V. Liu and M. Khalifa. Determinants of satisfaction at different adoption stages of Internet-based services. *Journal of the association for information systems* 4.1 (2003), 12.

[115] W. Maalej and D. Pagano. On the socialness of software. *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*. IEEE. 2011, 864–871.

[116] M. Maguire and N. Bevan. User requirements analysis. *IFIP World Computer Congress, TC 13*. Springer. 2002, 133–148.

[117] F. J. Massey Jr. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American statistical Association* 46.253 (1951), 68–78.

[118] F. Mäyrä. *An introduction to game studies*. Sage, 2008.

[119] J. A. McCall. Quality factors. *encyclopedia of Software Engineering* (2002).

[120] S. McIlroy, N. Ali, H. Khalid and A. E. Hassan. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empirical Software Engineering* 21.3 (2016), 1067–1106.

[121] T. Mens. Introduction and roadmap: History and challenges of software evolution. *Software evolution*. Springer, 2008, 1–11.

[122] T. Mens, M. Wermelinger, S. Ducasse, S. Demeyer, R. Hirschfeld and M. Jazayeri. Challenges in software evolution. *Eighth International Workshop on Principles of Software Evolution (IWPSE'05)*. IEEE. 2005, 13–22.

[123] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado and J. Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*. 2013, 3111–3119.

[124] A. Mockus, P. Zhang and P. L. Li. Predictors of customer perceived software quality. *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.* IEEE. 2005, 225–233.

[125] S. Moghaddam, M. Jamali and M. Ester. Etf: extended tensor factorization model for personalizing prediction of review helpfulness. *Proceedings of the fifth ACM international conference on Web search and data mining*. 2012, 163–172.

[126] S. M. Mudambi and D. Schuff. What Makes a Helpful Online Review? A Study of Customer Reviews on Amazon.com. *MIS Quarterly* 34.1 (2010), 185–200. ISSN: 02767783.

[127] S. Nadarajah. A generalized normal distribution. *Journal of Applied Statistics* 32.7 (2005), 685–694.

[128] V. Narayan. The Role of AI in Software Engineering and Testing. *International Journal of Technical Research and Applications* (2018).

[129] P. G. Neumann. *UK and Y2K: $50 billion*. `http://catless.ncl.ac.uk/Risks/19.07.html`. Accessed: 2021-03-27. 1997.

[130] M. E. Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences* 103.23 (2006), 8577–8582.

[131] H. A. Nguyen, A. T. Nguyen, T. T. Nguyen, T. N. Nguyen and H. Rajan. A study of repetitiveness of code changes in software evolution. *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE. 2013, 180–190.

[132] J. Nielsen and R. Molich. Heuristic evaluation of user interfaces. *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1990, 249–256.

[133] K. Nigam, A. K. McCallum, S. Thrun and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine learning* 39.2 (2000), 103–134.

[134] B. Nuseibeh and S. Easterbrook. Requirements engineering: a roadmap. *Proceedings of the Conference on the Future of Software Engineering*. 2000, 35–46.

[135] J. Oh, D. Kim, U. Lee, J.-G. Lee and J. Song. Facilitating developer-user interactions with mobile app review digests. *CHI'13 Extended Abstracts on Human Factors in Computing Systems*. 2013, 1809–1814.

[136] K. Okamoto, W. Chen and X.-Y. Li. Ranking of closeness centrality for large-scale social networks. *International workshop on frontiers in algorithmics*. Springer. 2008, 186–195.

[137] R. L. Oliver. Effect of expectation and disconfirmation on postexposure product evaluations: An alternative interpretation. *Journal of applied psychology* 62.4 (1977), 480.

[138] R. L. Oliver. A cognitive model of the antecedents and consequences of satisfaction decisions. *Journal of marketing research* 17.4 (1980), 460–469.

[139] R. L. Oliver. Measurement and evaluation of satisfaction processes in retail settings. *Journal of retailing* (1981).

[140] A. Oztekin, D. Delen, A. Turkyilmaz and S. Zaim. A machine learning-based usability evaluation method for eLearning systems. *Decision Support Systems* 56 (2013), 63–73.

[141] J. Paavilainen. Defining playability of games: functionality, usability, and gameplay. *Proceedings of the 23rd International Conference on Academic Mindtrek*. 2020, 55–64.

[142] D. Pagano and B. Bruegge. User involvement in software evolution practice: A case study. *2013 35th International Conference on Software Engineering (ICSE)*. IEEE. 2013, 953–962.

[143] D. Pagano and W. Maalej. User feedback in the appstore: An empirical study. *2013 21st IEEE international requirements engineering conference (RE)*. IEEE. 2013, 125–134.

[144] F. Palomba, M. Linares-Vásquez, G. Bavota, R. Oliveto, M. Di Penta, D. Poshyvanyk and A. De Lucia. User reviews matter! tracking crowdsourced reviews to support evolution of successful apps. *2015 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE. 2015, 291–300.

[145] F. Palomba, M. Linares-Vásquez, G. Bavota, R. Oliveto, M. Di Penta, D. Poshyvanyk and A. De Lucia. Crowdsourcing user reviews to support the evolution of mobile apps. *Journal of Systems and Software* 137 (2018), 143–162.

[146] Y. Pan and J. Q. Zhang. Born unequal: a study of the helpfulness of user-generated product reviews. *Journal of retailing* 87.4 (2011), 598–612.

[147] E. Papatheocharous and A. S. Andreou. Empirical evidence and state of practice of software agile teams. *Journal of Software: Evolution and Process* 26.9 (2014), 855–866.

[148] D. H. Park, M. Liu, C. Zhai and H. Wang. Leveraging user reviews to improve accuracy for mobile app retrieval. *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2015, 533–542.

[149] A. Perunicic. *Steam Scraper*. `https://github.com/prncc/steam-scraper`. 2018.

[150] D. Port, A. Nikora, J. H. Hayes and L. Huang. Text mining support for software requirements: Traceability assurance. *2011 44th Hawaii International Conference on System Sciences*. IEEE. 2011, 1–11.

[151] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto and D. Parisi. Defining and identifying communities in networks. *Proceedings of the national academy of sciences* 101.9 (2004), 2658–2663.

[152] B. Ramesh and M. Jarke. Toward reference models for requirements traceability. *IEEE transactions on software engineering* 27.1 (2001), 58–93.

[153] J. Ramos et al. Using tf-idf to determine word relevance in document queries. *Proceedings of the first instructional conference on machine learning*. Vol. 242. Piscataway, NJ. 2003, 133–142.

[154] F. Ricca, M. Torchiano, M. Di Penta, M. Ceccato and P. Tonella. Using acceptance tests as a support for clarifying requirements: A series of experiments. *Information and Software Technology* 51.2 (2009), 270–283.

[155] I. J. M. Ruiz, M. Nagappan, B. Adams, T. Berger, S. Dienst and A. E. Hassan. Examining the rating system used in mobile-app stores. *Ieee Software* 33.6 (2015), 86–92.

[156] P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering* 14.2 (2009), 131–164.

[157] N. S. Ryan, J. Pascoe and D. R. Morse. Enhanced reality fieldwork: the context-aware archaeological assistant. *Computer applications in archaeology*. Tempus Reparatum. 1998.

[158] A. R. Sampaio, H. Kadiyala, B. Hu, J. Steinbacher, T. Erwin, N. Rosa, I. Beschastnikh and J. Rubin. Supporting microservice evolution. *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE. 2017, 539–543.

[159] V. Sandulescu and M. Ester. Detecting singleton review spammers using semantic similarity. *Proceedings of the 24th international conference on World Wide Web*. 2015, 971–976.

[160] F. Sarro, M. Harman, Y. Jia and Y. Zhang. Customer rating reactions can be predicted purely using app features. *2018 IEEE 26th International Requirements Engineering Conference (RE)*. IEEE. 2018, 76–87.

[161] B. Schilit, N. Adams and R. Want. Context-aware computing applications. *1994 First Workshop on Mobile Computing Systems and Applications*. IEEE. 1994, 85–90.

[162] N. Seyff, F. Graf and N. Maiden. End-user requirements blogging with iRequire. *2010 ACM/IEEE 32nd International Conference on Software Engineering*. Vol. 2. IEEE. 2010, 285–288.

[163] N. Seyff, F. Graf and N. Maiden. Using mobile re tools to give end-users their own voice. *2010 18th IEEE International Requirements Engineering Conference*. IEEE. 2010, 37–46.

[164] G. W. Snedecor and W. G. Cochran. Statistical methods, 8thEdn. *Ames: Iowa State Univ. Press Iowa* 54 (1989), 71–82.

[165] D. S. Staples, I. Wong and P. B. Seddon. Having expectations of information systems benefits that match received benefits: does it really matter?: *Information & Management* 40.2 (2002), 115–131.

[166] Statista. *Mobile app usage - Statistics Facts*. `https://www.statista.com/topics/1002/mobile-app-usage`. Accessed: 2021-05-15. 2021.

[167] Statista. *Mobile application user retention rate worldwide from 2012 to 2019*. `https://www.statista.com/statistics/751532/worldwide-application-user-retention-rate/`. Accessed: 2021-05-15. 2021.

[168] Statista. *Retention rate on day 1 and day 30 of mobile app installs worldwide as of August 2020, by category*. `https://www.statista.com/statistics/259329/ios-and-android-app-user-retention-rate/`. Accessed: 2021-05-15. 2021.

[169] Statista. *Worldwide mobile app revenues in 2014 to 2023*. `https://www.statista.com/statistics/269025/worldwide-mobile-app-revenue-forecast/`. Accessed: 2021-05-15. 2021.

[170] J. Surowiecki. *The wisdom of crowds*. Anchor, 2005.

[171] A. Sutcliffe. *User-centred requirements engineering*. Springer Science & Business Media, 2002.

[172] A. Sutcliffe, S. Fickas and M. M. Sohlberg. Personal and contextual requirements engineering. *13th IEEE International Conference on Requirements Engineering (RE'05)*. IEEE. 2005, 19–28.

[173] M. Sutterer, O. Droegehorn and K. David. Upos: User profile ontology with situation-dependent preferences support. *First International Conference on Advances in Computer-Human Interaction*. IEEE. 2008, 230–235.

[174] J. E. Swan and I. F. Trawick. Satisfaction related to predictive vs. desired expectations. *Refining concepts and measures of consumer satisfaction and complaining behavior* (1980), 7–12.

[175] S. Syed and M. Spruit. Full-text or abstract? Examining topic coherence scores using latent dirichlet allocation. *2017 IEEE International conference on data science and advanced analytics (DSAA)*. IEEE. 2017, 165–174.

[176] B. Szajna and R. W. Scamell. The effects of information system user expectations on their performance and perceptions. *Mis Quarterly* (1993), 493–516.

[177] T. T. Tanimoto. IBM internal report. *Nov* 17 (1957), 1957.

[178] D. Tesch, J. J. Jiang and G. Klein. The impact of information system personnel skill discrepancies on stakeholder satisfaction. *Decision Sciences* 34.1 (2003), 107–129.

[179] M. Treiber, H.-L. Truong and S. Dustdar. On analyzing evolutionary changes of web services. *International Conference on Service-Oriented Computing*. Springer. 2008, 284–297.

[180] H. Van der Heijden. User acceptance of hedonic information systems. *MIS quarterly* (2004), 695–704.

[181] H. Van der Heijden, M. Ogertschnig and L. van der Gaast. Effects of Context Relevance and Perceived Risk on User Acceptance of Mobile Information Services. *ECIS*. Citeseer. 2005, 286–296.

[182] A. Van Lamsweerde. *Requirements engineering: From system goals to UML models to software*. Vol. 10. Chichester, UK: John Wiley & Sons, 2009.

[183] H. Van Vliet, H. Van Vliet and J. Van Vliet. *Software engineering: principles and practice*. Vol. 13. John Wiley & Sons Hoboken, NJ, 2008.

[184] L. Villarroel, G. Bavota, B. Russo, R. Oliveto and M. Di Penta. Release planning of mobile apps based on user reviews. *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. IEEE. 2016, 14–24.

[185] K. Vredenburg, J.-Y. Mao, P. W. Smith and T. Carey. A survey of user-centered design practice. *Proceedings of the SIGCHI conference on Human factors in computing systems*. 2002, 471–478.

[186] G. Wang, S. Xie, B. Liu and S. Y. Philip. Review graph based online store review spammer detection. *2011 IEEE 11th international conference on data mining*. IEEE. 2011, 1242–1247.

[187] Westwind. *Managing Customer Requirements: Requirements vs. Expectations*. `https://www.westwindconsulting.com/Requirements%20vs%20expectations.html`. Accessed: 2021-06-14. 2018.

[188] S. Wold, K. Esbensen and P. Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems* 2.1-3 (1987), 37–52.

[189] M. Xenos and D. Christodoulakis. Measuring perceived software quality. *Information and software technology* 39.6 (1997), 417–424.

[190] S. Yang, Y. Lu, S. Gupta and Y. Cao. Does context matter? The impact of use context on mobile internet adoption. *International Journal of Human-Computer Interaction* 28.8 (2012), 530–541.

[191] S. S. Yau, J. S. Collofello and T. MacGregor. Ripple effect analysis of software maintenance. *The IEEE Computer Society's Second International Computer Software and Applications Conference, 1978. COMPSAC'78.* IEEE. 1978, 60–65.

[192] R. K. Yin. Case study research: Design and methods, applied social research. *Methods series* 5 (1994).

[193] Z. Zainal. Case study as a research method. *Jurnal kemanusiaan* 5.1 (2007).

[194] A. Zerouali, E. Constantinou, T. Mens, G. Robles and J. González-Barahona. An empirical analysis of technical lag in npm package dependencies. *International Conference on Software Reuse*. Springer. 2018, 95–110.

[195] M.-L. Zhang and Z.-H. Zhou. ML-KNN: A lazy learning approach to multi-label learning. *Pattern recognition* 40.7 (2007), 2038–2048.

[196] W. Zhang and H. Mei. Software development based on collective intelligence on the Internet: feasibility, state-of-the-practice, and challenges. *SCIENTIA SINICA Informationis* 47.12 (2017), 1601–1622.

[197] Z. Zhang. Effective requirements development-A comparison of requirements elicitation techniques. *Software Quality Management XV: Software Quality in the Knowledge Society, E. Berki, J. Nummenmaa, I. Sunley, M. Ross and G. Staples (Ed.) British Computer Society* (2007), 225–240.

[198] T. Zimmermann, N. Nagappan and A. Zeller. Predicting bugs from history. *Software evolution*. Springer, 2008, 69–88.

[199] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter and C. Weiss. What makes a good bug report?: *IEEE Transactions on Software Engineering* 36.5 (2010), 618–643.

[200] I. A. Zolkepli, S. N. S. Mukhiar and C. Tan. Mobile consumer behaviour on apps usage: The effects of perceived values, rating, and cost. *Journal of Marketing Communications* (2020), 1–23.

# PUBLICATIONS

# PUBLICATION

# I

**A User-App Interaction Reference Model for Mobility Requirements Analysis**

X. Li and Z. Zhang

*International Conference on Software Engineering Advances (ICSEA)*, 2015, 170–177

# A User-App Interaction Reference Model for Mobility Requirements Analysis

Xiaozhou Li*, Zheying Zhang†
School of Information Sciences, University of Tampere
Email: *li.xiaozhou.x@student.uta.fi, †zheying.zhang@uta.fi

***Abstract*—Contemporary mobile applications (apps) value mobility as a key characteristic that allows users to access services or features ubiquitously. In order to achieve decent mobility, apps shall provide features that are suitable to use under a wide range of contexts. In this paper, we analyze the situational contexts, towards which the mobile apps shall comply with in terms of mobility. By analyzing the contexts and the ways of interaction between users and apps, we propose and illustrate a mobile requirements analysis process model to identify the conflicts between users' ideal ways of interaction and the way the feature is designed to provide. The identified conflicts help to elicit requirements for the enhancement of the apps' mobility.**

***Keywords–Mobility; Mobile application; Requirements; Context; Situational Context; Interaction;***

## I. Introduction

The emergence of iOS and Android OS has been changing the mobile industry and people's daily lives, and been providing new trends in the academic research [1]. Changes in distribution process and mobile software market mechanism lead to better customer accessability towards mobile apps and inevitable competition [2]. The ranking mechanism also intensifies the competition, demanding mobile apps satisfying users' diversified demands, which shall be reached in varying situations, compared to desktop software. It thus requires companies to take into account the capability of the mobile applications to provide satisfactory user experience regardless the changing environment [3].

Mobility is one of the most significant and unique features for mobile apps, refering to the ability to access services ubiquitously through wireless networks and various mobile devices [4][5]. The vision of mobility is to be able to work "anytime, anywhere" [6]. However, with limited support of systems towards mobility, the capability of being comfortably used at "anytime, anywhere" of mobile apps is seldom achieved. Thus, achieving mobility shall result in the enhanced competitiveness of mobile apps in terms of user satisfaction.

Contexts, which refers to the information that characterizes the situation of an entity, has a great impact on usability and user experience of mobile apps [7]–[10]. Mobility, as a key aspect of usability of mobile apps, is also affected largely by their context [11], which also influences the way, by which a user interacts with an mobile app [12]. By specifying the ways, in which user and app interact, we analyze their relation towards different contexts. The elicited mobility requirements shall thus reflect suitable interaction ways between users and apps in differen contexts.

Many studies analyze the phenomena of use situations concerning mobile commerce [4][13] and other types of mobile apps [14]. But studies on mobility requirements analysis are very limited. A goal-oriented framework for modeling and analyzing requirements for varying contexts was proposed based on the goal model to reason variants [15]. But it fails to address how to identify the varying contexts and derive requirements regarding the way of interaction. Several other studies [16][17] address challenges and methods of requirements analysis for mobile systems and pervasive services, but lack a concrete proposal on taking varying contexts into account for requirements analysis.

In this study, we focus on the analysis of context of use of mobile apps and the possible ways of interaction between users and an app, and further study the way of analyzing mobility requirements. The paper tries to tackle the following questions.

- RQ1 *What are the contexts that affect the interaction between user and mobile apps?*
- RQ2 *What are the ways in which a user interacts with mobile apps, and what are their relations with different contexts?*
- RQ3 *How to take into account contexts and ways of interaction when analyzing apps mobility requirements?*

The purpose of this research is to enhance the mobility of mobile apps by taking into account the varying contexts in requirement analysis. To answer RQ1, we summarize the definition of mobility and main perspectives of mobile app contexts by reviewing the literature on the concept of mobility and context in Section 2. We also tackle RQ2 by analyzing the relation between the way users interact with mobile apps and different contexts via user-app interaction reference model in Section 3 and 4. In Section 5, we propose our approach to analyzing mobility requirements with a case study for further illustrating and discussing in Section 6, which altogether answers RQ3. Section 7 concludes with implications for future research.

## II. Mobile Apps and Their Mobility Requirements

Mobile devices and applications has enabled new freedom and flexibility on the way people communicate, work, and entertain by providing services beyond the constraints of fixed locations and devices. Compared to the old style of using manufacturer provided mobile software, contemporary mobile apps have lower distribution costs and can be more easily accessed by customers via the change in distribution process and mobile software market mechanism [2]. The mechanism stimulates the development of mobile application markets and results in fierce competition with even software on personal computers challenged. Prior to the investigation of the circumstances, under which users would prefer to mobile applications rather than desktop software, the unique mobility characteristics of mobile applications differentiated from those of desktop software shall be understood.

### A. Mobility

Mobility was understood as the human's independency from geographic constraints or the ability and/or quality to ensure the given entity can move or be moved [18]. Mobility primarily facilitates a mobile device to operate properly when

 ISBN: 978-1-61208-438-1 

its location changes. This provides a generic view of how mobility is supposed to be acquired by users when they use mobile apps, and forms a general goal in mobile app development. According to [5], the key feature of mobile technology is the capability of using services on the move, with wireless network and various devices, which provides the literal meaning of mobility. Other similar terms, such as nomadicity, which indicates a system's capability of providing services to the nomad as he moves from place to place in a transparent and convenient form [6], also reflexes the concept of mobility.

Consequently, mobility is an attribute of both human beings and the computational devices they interact [11]. The mobility of mobile devices refers to the ability to access services ubiquitously, or "anytime, anywhere" through wireless networks and various mobile devices [4][6] and for mobile apps as well. As a critical feature of mobile apps usability, mobility has considerable impact on the interaction between users and mobile devices and apps [19]. Thus, compared to mobility of human beings, mobile app mobility is seen as the usefulness and ease of use provided by the app towards user satisfaction in "anytime, anywhere".

There are three types of mobility in terms of modality [20], i.e., travelling, visiting and wondering. The mobility towards the usability of mobile apps is hence seen in these three perspectives as well, that is, to provide services when users are traveling, visiting, and wondering. Similar categorization is also given by [21], which describes the motion of mobile app users into none motion, constant motion and varying motion. Besides the categorization of mobility regarding spatial movement, time and context changes also contribute to the mobility attribute provided by mobile apps [18]. In this study, we see all the factors that influence the mobility of the mobile app from outside the app itself as its context.

From the context of use perspective, mobility implies that the app shall provide context-aware features and/or services. Following the definition of the concept of context, i.e., the information that can be used to characterize the situation of an entity [7][8], where an entity can be a place, person, physical or computational object, we define context-aware features as the use of context to provide task-relevant information and/or features, which a user feels easy to use. The situation can be characterized in perspectives, such as location, surrounding changing objects, and people, and can define where you are, who you are with, and what resources are nearby [22][23]. These perspectives can be further refined into users, tasks, equipment (i.e., hardware, software and materials), location, physical environment, temporal context, social environment, technical and information context, etc. and have been intensively addressed and adapted in surveys and research on the context in mobile computing and the impact on the overall design of the product [8][10][17][24]–[28] .

### B. Mobility Requirements Analysis

We consider mobility as an intrinsic attribute of mobile applications. It refers to the capability of providing receptive and pleasant services acquired by users in spite of the changes in environments. Such an attribute can be refined into different types of requirements contributing to users' satisfaction. The requirements include functional requirements complementing the main features of an application and supporting users to fulfill their goals, interface requirements facilitating the interaction between users and the application, as well as constraints on the application.

In addition to analyzing the core features of an application, mobility requirements analysis shall emphasize ease of use in the dynamic environment of use, and focus on analyzing the diversity of context of use and ways of interaction between users and the app. Accordingly, we adapt the generic requirements syntax of Mavin et al.'s EARS model [29] for mobility requirements, emphasizing the context and the interaction with users, as shown below.

**In <situational contexts>,<optional preconditions> <optional trigger> the <mobile app name> shall <app response> in <ways of interaction>.**

The syntax marked in grey is what was specified in the EARS model [29]. It can be further specialized into different types of requirements following temporal logic defined between the precondition, the trigger, the app response, etc. [29]. In addition, the components marked in black are situational contexts and ways of interaction, which highlights the mobility attributes a mobile app shall reflect and the requirements analysis shall take into account. The situational context encompasses a wide range of elements, such as the location and surroundings, the social context, the user's movement, the temporal context, etc. Combining value of these elements forms a variety of scenarios of using a mobile app. Changes of the scenarios continuously reframe a user's interaction with a mobile app. Obviously not all scenarios are desired and friendly. The requirements analyst shall be aware of the suitable ways of interaction is adopted towards typical scenarios, which secures users' satisfaction and receptiveness largely.

## III. A User-App Interaction Reference Model

Interaction between a user and a mobile app occurs after the user opens the app and before he or she closes it [30]. We call it a user-app interaction. According to the definition of context given by [7][8], the context of a user-app interaction is referred to as the information to characterize the situation of the two entities, i.e., the user and the app. The context is further depicted in Figure 1.

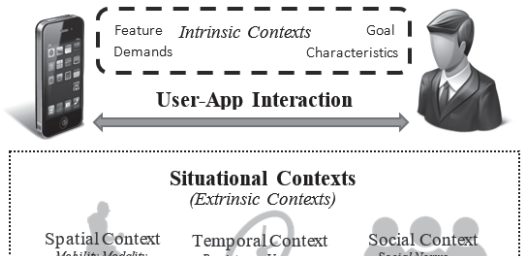


Figure 1. User-App Interaction Reference Model.

The context shall ideally contain all possible situations that affect the user-app interaction. Similar to the contexts categories described in previous studies [17][22]–[24], the ideal context shall contain multiple perspectives including user profile, operating system, hardware system and network, physical context, temporal context, task context, and social context. As shown in Figure 1, we divide the wide range of context into intrinsic and extrinsic ones. The intrinsic context refers to the inner attributes of entities that influence, or occasionally determine the occurrence of a user-app interaction. For example, the features provided by mobile apps and demands for operating them intrinsically determine their usefulness when users' goals and their characteristics determine whether to use. The extrinsic context refers to the external factors that influence a user's decision of his or her engagement in a user-app interaction. The extrinsic contexts, defined also as situational contexts, include device context, environmental context, spatial context, temporal context and social context. The device context (e.g. system, network, hardware, etc.) and the environmental context (e.g. light, noise, wind, temperature, etc.) have been often studied in requirements engineering for self-adaptive software systems [31]. However, the other extrinsic contexts, such as spatial, temporal, and social contexts are rarely discussed in requirements analysis process. Hereby, we focus on these situational contexts to investigate their relations with the mobility attribute of a mobile app.

### A. Intrinsic Contexts

Many researchers have addressed and verified that a user's demographic properties, such as age, gender, education, income, etc. are relevant factors affecting a user's attitudes and preference to the use of an app [17][28][32]–[34]. Besides the demographic properties, individual users with various interests and attitudes toward the mobile value form other important perspectives influencing a user's engagement in an app activity. Users' adoption to mobile services has been analyzed from the perspectives of a user's characteristics, major value, attitude, and major interests. [28]. Concerning a user-app interaction, the user's goals refer to the objectives of the user at the critical moment when the interaction occurs. For an instance, the goal of a student working on an exam is to pass the exam. It is not only related to mobile applications but also has a wider range than the term described in goal-oriented requirements engineering [35]. Thus the characteristics and goals of the user form the intrinsic and determinant context to the initiation of a user-app interaction [36].

On the other hand, a user-app interaction occurs when a user determines to perform a task associated with a mobile app with a particular purpose, i.e., fulfilling a user's goal. The features of a mobile app contain capabilities that enables users to fulfill their goals by accomplishing the tasks. The user is prone to intrinsically start an interaction with the app when the provided features comply with his or her intention to achieve his or her goal. The demands are defined as the workload that a user is obliged to engage in order to accomplish the task as the demands, which contain six subscales, i.e., mental demand, physical demand, temporal demand, frustration, effort and performance [37]. They define the subjective experience of users on using the app and are affected by the intrinsic contexts of a user-app interaction as well.

### B. Situational Context Model

The situational context refers to the extrinsic properties of the user and the app that impact the initiation of a user-app interaction. As mentioned above, we only focus on the temporal, spatial and social perspectives to discuss the situational context in this study.

*1) Temporal Context:* Temporality, as one of the dimensions of the mobility concept originally [18], has been influenced by the mobile technology inherently in terms of human interaction. The multiple perspectives of temporality, such as structural and interpretive, monochronicity and polychronicity and so on have been studied previously [18][38]. Compared to the previous frameworks, we argue that the temporality in terms of a user-app interaction is determined by the user's sense of time and the persistence of the app to accomplish one operation session, and define temporal context in this paper as the sense of external time pressure of the user caused by the confliction or the accordance of user's goal and app's demands. Thus, two values of intensive and allocative are used to describe temporal context. Intensive refers to the situation when the user is in urgent need of achieving his or her goal and has limited spare time of interacting with the app (e.g. the user is busy in working on assignments with approaching deadline). Allocative, on the other hand, indicates that the user has no urgent goal to achieve and is temporally available (e.g. the user is staying at home idle).

*2) Spatial Context:* The spatial perspective of mobility indicates the geographic movement of the user when engaged in the interaction with the mobile app [18]. In this study, the spatial context refers to the current movement of the user further indicating the physical availability for the app usage. we adopt the mobile modality types given by [20] categorizing the spatial context, including visiting, traveling and wandering. Visiting context indicate that the user is in a physically stationary status(e.g. sitting in a meeting). Traveling context refers to the situation when the user is in a transportation tool (e.g. a car or train). Wandering, on the other hand, refers to the situation when the user is physically moving from place to place (e.g. walking or running). However, the categorization given by [20] did not specify the difference between driving a transportation tool or sitting in one in terms of traveling perspective. In addition, exercise related scenarios of walking or running is not taken into account either. In this study, these distinctions shall be reflected by the combination with other contexts.

*3) Social Context:* The social context is interpreted by [8][10][17][24]–[28][39] as the influence of other persons' presence and the interpersonal interaction between the user and others. In this study, we interpret the social context of a user-app interaction as the social norms that constrain user from or encourage user into the interaction [40], which contains similar meaning towards the functional place concept in [39]. We define the scale of social context from constraining to encouraging the use of apps based on the social norms. For example, a conference presentation is socially constraining when idleness at home is socially encouraging.

According to the three perspectives of situational context mentioned above, values are assigned to each perspective, combining which leads to a unique context scenario description (shown in Table I).

Ideally, based on the given situational context model, the situational context of user can be described by the combination

TABLE I. VALUES OF SITUATIONAL CONTEXT PERSPECTIVES.

| *Perspective* | *Value* |
|---|---|
| Temporal | **I**ntensive, **A**llocative |
| Spatial | **V**isiting, **T**raveling, **W**ondering |
| Social | **C**onstraining, **E**ncouraging |

of the three perspectives. The 12 situational contexts include *Intensive-Visiting-Constraining (IVC), Allocative-Visiting-Constraining (AVC), Intensive-Visiting-Encourage (IVE), Allocative-Visiting-Encourage (AVE), Intensive-Traveling-Constraining (ITC), Allocative-Traveling-Constraining (ATC), Intensive-Traveling-Encouraging (ITE), Allocative-Traveling-Encouraging (ATE), Intensive-Wondering-Constraining (IWC), Allocative-Wondering-Constraining (AWC), Intensive-Wondering-Encouraging (IWE), and Allocative-Wondering-Encouraging (AWE)*. For each combination of values from different perspectives, we provide a typical situational context scenario, shown in Table II.

TABLE II. TYPICAL SCENARIOS FOR EACH SITUATIONAL CONTEXT.

| *Situational Context* | *Typical Scenario* |
|---|---|
| IVC | In a conference giving presentation |
| AVC | In a lecture listening |
| IVE | In a cafe working on assignments with close deadline |
| AVE | At home idle |
| ITC | In a car driving with time limit |
| ATC | In a car driving and sight seeing |
| ITE | In a train when it is about to arrive at the destination |
| ATE | In a train idle |
| IWC | Running in a race |
| AWC | Wondering in a cocktail party as a host |
| IWE | Running to catch a bus |
| AWE | Walking in a park relaxing |

In practice, the scenarios that used for describing situational contexts might vary based on the collective understanding of the contexts from the team. For example, the scenario "In a conference giving presentation" and "in a contract signing meeting negotiating" can both be used describing the situational context of IVC.

## IV. WAYS OF USER-APP INTERACTIONS

The concept of mobility is not only just a matter of people traveling, but also the interaction people perform, that is, the way in which they interact with each other [18]. The mobility is thus reflected in the way in which users interact with the apps. It occurs when an app sends out a notification to the user who responds it and ends when the user finishes using the app and closes it. However, users in different situational contexts, who have different goals and characteristics, will expect to interact with different features of the app differently but comfortably. In order to find the match between the designed and expected ways of interaction, we adapt the dimensions of interaction modality [41][42] discussing the situational characteristics of mobile apps including their obtrusiveness and persistence.

An obtrusive interaction imposes obligation to notice or react [18], which indicates that the interaction is evoked by notifying the user to start it without the user's internal motivation to do so. For example, an obtrusive interaction is initiated when the user stops original reading activity and responds to the new message notification from WeChat. On the contrary, an unobtrusive interaction is initiated with the user's internal motivation. For example, the user encounters an unfamiliar term while reading and decides to look it up in Eudic without receiving notification. On the other hand, the persistence dimension specifies the duration of an interaction, which is largely depending on the time length a user spends on completing an interaction task. An ephemeral interaction requires a short time to achieve user's goal (e.g. replying a message, looking up a word). A persistent interaction oppositely takes a long period to accomplish (e.g. playing Subway Surfers, listening to Spotify).

With the two dimensions combined, a user-app interaction can thus be described as *obtrusive-persistent (OP), unobtrusive-persistent (UP), obtrusive-ephemeral (OE), or unobtrusive-ephemeral (UE)*. By analyzing the relation between different types of user-app interactions and the way they fit in the process of the way of the user's original activity, we conclude four ways of interaction, including *intermittent*, *interrupting*, *accompanying*, and *ignoring*.

An intermittent way of interaction refers to the interlaced engagement in both the user's original task and the user's interaction towards the mobile app, with the whole process of several short interactions, which are neither consistent nor interfering the proceeding of the original task. For example, when watching TV, the user starts the interaction with WeChat. Within the whole process, the user inconsistently responds messages but his or her task of watching TV remains proceeding. An intermittent way of interaction often consists of a number of ephemeral interactions, which are also mostly obtrusive.

An interrupting way of interaction requires the user to convert full concentration on the interaction and cease the original activity. For example, to start playing Subway Surfers, the user has to stop the original task, such as reading books or watching TV. It can be interpreted as the original task is interrupted by this user-app interaction. An interrupting way of interaction is mostly persistent.

An accompanying way of interaction refers to the paralleling engagement in both user's original task and the user-app interaction tasks. Comparing to the interrupting or the intermittent way of interaction, the accompanying one will not attract the user's full attention, as the user does not stop the continuous progress of the original task. For example, when running on a treadmill, the user starts watching films from Netflix. The activity of running is, instead of interrupted, paralleling with the user-app interaction. The interaction can be ephemeral or persistent, depending on the amount of engagement an app requires from the user.

An ignoring way of interaction indicates that the interaction with the mobile app is ignored by the user in order to maintain the proceeding of his or her original task. For example, when taking an examination at school, the user will ignore any types of interaction with the mobile apps.

The relation between different types of user-app interactions and the according ways of interaction is summarized in Figure 2. In Figure 2, the narrow arrows underneath represent the user's original task, and the thick ones represent the user-app interactions. Lighter gray arrows are the interactions ignored and not executed. In addition, the length of the arrows indicates the timeline of proceeding with the task or interaction.

By analyzing the different ways of user-app interactions, we are enabled to analyze the expected way of interactions towards each mobile app feature. And towards mobility, expected ways of interaction shall comply with the previously

 ISBN: 978-1-61208-438-1 

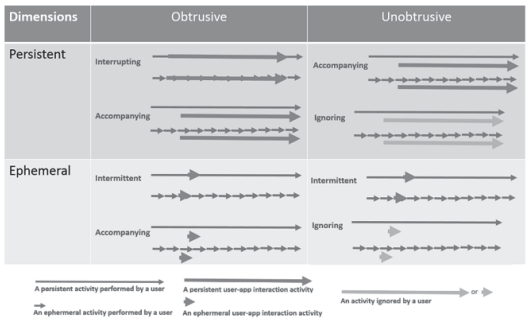


Figure 2. Ways of User-App Interactions

defined situational contexts. Combining this analysis, we shall be able to detect how an app feature is expected to perform in different situational contexts. For requirements analysts, each feature and the refined requirements could be analyzed and mapped into the situational contexts assigned with expected ways of user-app interactions. For example, Table III shows the expected ideal ways of interaction towards given situational context .

TABLE III. IDEAL WAY OF INTERACTION FOR EACH SITUATIONAL CONTEXT.

| *S.C.* | *Typical Scenario* | *Ideal Ways* |
|---|---|---|
| IVC | In a conference giving presentation | Accompanying |
| AVC | In a lecture listening | Intermittent |
| IVE | In a caf working on assignments with close deadline | Intermittent, Accompanying |
| AVE | At home idle | Interrupting Intermittent Accompanying |
| ITC | In a car driving with time limit | Accompanying |
| ATC | In a car driving and sight seeing | Accompanying, Intermittent |
| ITE | In a train when it is about to arrive at the destination | Accompanying, Intermittent |
| ATE | In a train idle | Interrupting, Intermittent Accompanying |
| IWC | Running in a race | Accompanying |
| AWC | Wondering in a cocktail party as a host | Intermittent |
| IWE | Running to catch a bus | Accompanying |
| AWE | Walking in a park relaxing | Accompanying, Intermittent |

Table III indicates that in a specific situational context one mobile app feature shall enable users to interact comfortably in the ideal ways of interaction. Taking the situational context scenario of IVC as an example, the ideal way is the accompanying way of interaction. Thus, a mobile app feature which offers such a way of interaction is more likely to be used in this circumstance. For example, the slide presentation feature of Prezi can provide accompanying way of interaction in the IVC context scenario of "In a conference giving presentation". The typical scenario of situational contexts can be also different. For example, IVC context can also be represented in the scenario of "In a university exam with time limit" or "In a chess competition with time limit for each move". When a certain feature fails to initiate the ideal ways of interaction, it shall be adjusted at requirement specification level towards the ideal ways. Therefore, a process of identifying such features and specifying the according strategy of mobility enhancing adjustment is required.

## V. MOBILITY REQUIREMENTS ANALYSIS PROCESS

The mobility requirements analysis process contains a sequence of pre-defined steps, by following which requirements analysts can specify existing user requirements towards enhanced mobility. The aim of mobility requirements analysis is to provide specified requirements that enable users to use the given features in a satisfied way in the possible situational contexts. As defined previously, a user's satisfaction for a specific feature is achieved by using this feature in different situational contexts via ideal ways of interactions. Thus the proposed mobility requirements analysis process is to refine existing app features by taking into account the given situational contexts and the according ways of interactions. The process of mobility requirements analysis is described as Figure 3.

```
Input: Requirements Document (RD)
Output: Mobility enhanced Requirements Document (MERD)

Identify Primary Situational Contexts
for each feature in RD, do
        Specify Interaction
        for each specified interaction, do
                for each primary situational context, do
                        if way of interaction ∉ the ideal way of interaction,
                        then change the requirements towards primary situational context;
                             document the requirements and changes into MERD;
return MERD
```

Figure 3. Mobility Requirements Analysis Process.

The analysis process consists of four key steps, as explained below.

*Step 1. Identify the Primary Situational Contexts*

Mobile apps are meant to satisfy users' needs in all possible situational contexts in an ideal way. However, mobile apps have distinct visions and features, and cannot comply with every situational context to meet users' needs. It thus requires the requirements analysts to identify the primary situational contexts by prioritization. The outcome of this activity is a list of prioritized situational contexts, or a number of primary situational contexts.

*Step 2. Specify the Expected Way of Interaction for Each Feature*

For each feature of the mobile app, requirements analysts shall be able to specify an expected way of interaction, which is expected by users. It means that users will use this feature most comfortably via that way of interaction. The outcome of this activity is a list of features together with expected ways of interaction respectively.

*Step 3. Compare the Previous Two Outcomes and Identify the Conflicts*

By comparing the outcomes of the previous steps with Table III, we can find the features, of which the expected way of interaction, conflicts with the ideal ones of the primary situational contexts. These conflicts shall be adjusted in the next step to enhance the app's mobility attribute.

*Step 4. Adjust Conflicting Feature towards Mobility Requirements*

We diminish the conflicts by changing the requirements related to the feature or adding new ones.

## VI. CASE STUDY

By following the steps of mobility requirements analysis process, we are able to identify the features of a mobile app that may contain conflicts against the ideal ways of

 ISBN: 978-1-61208-438-1 

interaction in specified primary situational contexts, and also to analyze and adjust the according features towards eliminating the conflicts, hence enhancing their mobility. In this section, we apply our proposed approach to analyzing three mobile apps, WeChat[43], Gmail[44], and AlienBlue[45]. WeChat is a messaging and calling app. It allows users to communicate with friends for free text (SMS/MMS), voice & video calls, moments, photo sharing, and games. Gmail (IOS) is the official mobile app for iPhone and iPad. It supports real-time notifications of new mails, multiple accounts, and mail search across the entire inbox. AlienBlue is the official app for Reddit, an online bulletin system. It enables users to browse threads from Reddit, post new threads and reply on others' threads with other features, such as liking or disliking, subscribing, image uploading, and so on.

The three mobile apps share the essential feature of user communication but contain differences in details. For example, WeChat enables users to receive, send, and share multimedia messages instantly. Gmail contains no voice messaging feature and takes longer time on individual operation session, such as browsing and replying emails. On the other hand, the communication between users on AlienBlue is fulfilled by posting, reading and replying threads in Reddit. Thus, in the study, we focus on the communication feature of the three apps to analyze their mobility attributes and requirements.

*Step 1. Identify the Primary Situational Context*

Firstly, the primary situational contexts shall be identified amongst the previously defined 12 situational contexts, as well as the scope of the analysis. Instead of prioritizing the 12 situational contexts precisely, for these cases, we categorize situational contexts into three prioritization level, including, *primary*, *secondary* and *ignorable*. Primary situational contexts indicates that most users tend to use this feature in these situational contexts. Secondary contexts are those situational contexts in which the user has the equal possibility of using the app or not. And ignorable contexts are those in which users nearly never use the feature. In terms of the "user communication" feature of the three cases, the according categorization of situational context is shown as Table IV.

TABLE IV. PRIMARY SITUATIONAL CONTEXTS.

| | *WeChat* | *Gmail* | *AlienBlue* |
|---|---|---|---|
| Primary | AVE, ATE, AWE | AVE, ATE | AVE, ATE |
| Secondary | AVC, IVE, ITE, AWC | ITE, AWE | AWE |
| Ignorable | IVC, ITC, ATC, IWC, IWE | IVC, AVC, IVE, ITC, ATC, AWC, IWC, IWE | IVC, AVC, IVE, ITC, ATC, ITE, IWC, AWC, IWE, |

Taking WeChat as an example, it enables users instant communication. Thus, the actions of reading and replying messages is to a large extent encouraged in the situation without social constraints (e.g. driving for safety reason) or time limit towards other objectives (e.g. deadlines). The social encourging and time allocative contexts are also the primary contexts for the other two apps. But different from them, instant communication feature of WeChat is also encouraged in 'wondering' contexts. Besides, even with certain social constraints and time limits, users tend to use WeChat more than the other two, which is why more secondary situational contexts are identified for WeChat.

*Step 2. Specify the Expected Ways of Interaction*

The expected way of interaction for the feature shall be determined by the way in which most of the users use the feature, which can be identified and analyzed by using different requirements elicitation techniques, or asserted by requirements analysts based on the use pattern of other similar products. For example, the expected way of interaction for WeChat communication is *intermittent*, as users only allocate short time for instant communication without original activity fully interrupted. Comparatively, Gmail and AlienBlue require more concentration and time from users for reading and replying emails, which results in an *interrupting* way of interaction.

*Step 3. Compare the Previous Two Outcomes and Identify the Conflicts*

Comparing the pre-defined ideal ways of interaction for the primary situational contexts and the expected way of interaction for the feature, we find no conflicts for all apps in their primary situational contexts (shown in Table V). When no conflicts are found for all primary situational contexts, we can indicate that this existing feature provides adequate mobility support.

TABLE V. COMPARISON OF IDEAL AND EXPECTED WAYS OF INTERACTION

| *Primary* | *Ideal* | *WeChat* | *Gmail* | *AlienBlue* |
|---|---|---|---|---|
| AVE | Interrupting, Intermittent, Accompanying | Intermittent | Interrupting | Interrupting |
| ATE | Interrupting, Intermittent, Accompanying | Intermittent | Interrupting | Interrupting |
| AWE | Intermittent, Accompanying | Intermittent | **Interrupting** | **Interrupting** |

However, conflicts are found in the secondary context of AWE for Gmail and AlienBlue. Compared with the primary contexts, we find that users in "wondering" context are more likely to start interaction with WeChat rather than Gmail and AlienBlue based on their expected ways of interaction. Thus, to enhance the mobility of them, conflicts for this secondary situational context shall be addressed with additional mobility requirements as in practise the secondary situational contexts might be also of high priorities.

*Step 4. Adjust Conflicting Feature towards Mobility Requirements*

Once the conflicts were detected, the according feature or function shall be adjusted in order to improve the overall mobility of the app. According to the conflicting situational context (i.e., AWE), the mobility requirement to adjust is as follows.

*In a situational context of "Allocative-Wandering-Encouraging", the Gmail/AlienBlue app shall provide user text-based communication functionality in the intermittent way of interaction.*

The mobility requirements provide the goal for requirements analysts indicating which specific situational contexts and by which ways of interaction the target feature shall be adjusted. The adjustment can be applied by adding or editing the existing requirements related to this function. Taking AlienBlue as an example, part of functions related to text-based communication in thread discussion is summarized in Table VI.

When adjusting the functions, we shall take into account the conflicting situational context. The perspective that plays a critical part of the conflict is firstly focused. For example, concerning the specific situational context of "Allocative-

 ISBN: 978-1-61208-438-1 

TABLE VI. ALIENBLUE'S FUNCTIONS

| *App Name* | *Functions* |
|---|---|
| AlienBlue | AFR1.The app allows the user to view through the whole thread;<br>AFR2.The app allows the user to reply on a specific comment;<br>AFR3.The app allows the user to send replies with images and emojis;<br>AFR4.The app allows the user to like or dislike other comments;<br>AFR5.The app allows the user to receive comments notifications; |

Wandering-Encouraging", social encouraging and time allocative contexts do not hinder user's interaction with the thread receiving and replying feature of AlienBlue. Thus, within the range of this very feature, we change the existing function with more specified function variations. As follows, based on the given functions, we provide examples on how to change the existing function or add new functions that comply with the "Wandering" situational context.

Taken as examples, AFR1 and 2 are two of the essential function of the AlienBlue app, which must not be removed. However, as a persistent and non-obtrusive functions, based on Figure 2, these functions are prone to be ignored in most situational contexts, especially for the "Wandering" context. One way to change them is to shorten the operating session.

Thus, the AFR1 and AFR2 can be changed into:

AFR1.1 *The app shall allow the user to view exclusively his or her own comments and the ones he or she comments on with unrelated comments folded;*

AFR1.2 *The app shall allow the user to view unrelated comments by unfolding them;*

AFR1.3 *The app shall allow the user to quickly control the display of the thread interface by hand gestures;*

AFR1.4 *The app shall allow the user to view comments concerning him or her on the lock screen;*

AFR2.1 *The app shall allow the user to reply with predefined quick responses;*

AFR2.2 *The app shall allow the user to save unfinished comments automatically and to continue composing;*

AFR2.3 *The app shall allow the user to respond to the received comments on the lock screen;*

Compared to the originial requirements, the specified requirements largely reduced the browsing time by directly enabling the user to focus on the relevant comments. Meanwhile, the specified requirements also enhance the obtrusiveness of the notification, which allows the user to better respond to the notification. In this way, based on the existing functions, these functions are adjusted in order to eliminate the conflicts between ideal way of interaction in a certain primary situational context and the expected way of interaction of this feature. By repetitively doing so with all the features of the mobile app, the mobility of the target mobile app is supported as users are enabled to interaction with the features in the expected way of interaction.

In this case study, we adopt existing mobile apps as examples to demonstrate how the mobility requirements analysis process can be applied. It is easy to identify the features that require mobility enhancement and the corresponding change proposal for well-known apps. In practise, it is hard to predict and fully specify all situational contexts and assure that the target app attract users to use in their expected way. The proposed approach and process provides a way of analyzing situational contexts, in which the app is put to use and eliciting requirements that enhance its mobility. This study contributes in filling the gap in the studies on applying the understanding of context into mobile app requirements analysis. Furthermore, the user-app interaction reference model and the situational context analysis provides an extensible framework of studying the context of a user-app interaction where more perspectives can be added to enrich the scenario set of situational contexts. This approach also enables developers to choose the suitable set of context scenarios and prioritization, as well the ideal ways of interactions, based on the vision and scope of their target mobile apps.

## VII. CONCLUSION

In this paper, we explore the concept of mobility as the characteristic of mobile apps, which satisfies users' need to use them under changing contexts. By analyzing the perspectives of mobility, we define situational contexts as the key extrinsic factors that influence users' satisfaction in user-app interactions. Compared to the other context factors, such as device context and environmental context, the situational contexts are more tangible towards the understanding of how users and apps interact, and also the factors shall be taken into account when mobile development team aims to enhance the mobility of their mobile products.

Furthermore, based on the specification of typical situational context scenarios, we further analyze connection between these situational contexts and the ideal ways of user-app interactions. Hence, seeking the conflicts between ideal ways of interaction and the current ones is the method to detect the key mobility-lacking features of a mobile app. On the basis of the analysis, we propose the mobility requirements analysis process, which helps to adjust features and the according requirements towards the ideal ways of interaction. Accordingly, the overall mobility of the mobile app improves.

The future work of this study will focus on the other extrinsic contexts and their influences on user-app interactions, which shall be utilized as the replenishment for the existing reference model. The user characteristics and goals, as well as their connection towards the mobile application feature and demands, shall also be reviewed and analyzed in the mobile app domain. In addition, the connection between the improvement of mobility and user satisfaction to mobile apps shall be also studied as the validation of our mobility requirements analysis method in our future studies.

## REFERENCES

[1] D. Gavalas and D. Economou, "Development platforms for mobile applications: Status and trends," Software, IEEE, vol. 28, no. 1, 2011, pp. 77–86.

[2] A. Holzer and J. Ondrus, "Mobile application market: A developers perspective," Telematics and informatics, vol. 28, no. 1, 2011, pp. 22–31.

[3] C. Ryan and A. Gonsalves, "The effect of context and application type on mobile usability: an empirical study," in Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38. Australian Computer Society, Inc., 2005, pp. 115–124.

[4] N. Mallat, M. Rossi, V. K. Tuunainen, and A. Öörni, "The impact of use situation and mobility on the acceptance of mobile ticketing services," in System Sciences, 2006. HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on, vol. 2. IEEE, 2006, pp. 42b–42b.

[5] C. Coursaris and K. Hassanein, "Understanding m-commerce: a consumer-centric model," Quarterly journal of electronic commerce, vol. 3, 2002, pp. 247–272.

[6] L. Kleinrock, "Nomadicity: anytime, anywhere in a disconnected world," Mobile networks and applications, vol. 1, no. 4, 1996, pp. 351–357.

 ISBN: 978-1-61208-438-1 

[7] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in Handheld and ubiquitous computing. Springer, 1999, pp. 304–307.

[8] A. K. Dey, "Understanding and using context," Personal and ubiquitous computing, vol. 5, no. 1, 2001, pp. 4–7.

[9] L. Barnard, J. S. Yi, J. A. Jacko, and A. Sears, "Capturing the effects of context on human performance in mobile computing systems," Personal and Ubiquitous Computing, vol. 11, no. 2, 2007, pp. 81–96.

[10] H. Korhonen, J. Arrasvuori, and K. Väänänen-Vainio-Mattila, "Analysing user experience of personal mobile products through contextual factors," in Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia. ACM, 2010, p. 11.

[11] L. Gorlenko and R. Merrick, "No wires attached: Usability challenges in the connected mobile world," IBM Systems Journal, vol. 42, no. 4, 2003, pp. 639–651.

[12] A. Oulasvirta, S. Tamminen, V. Roto, and J. Kuorelahti, "Interaction in 4-second bursts: the fragmented nature of attentional resources in mobile hci," in Proceedings of the SIGCHI conference on Human factors in computing systems. ACM, 2005, pp. 919–928.

[13] C. Kim, M. Mirusmonov, and I. Lee, "An empirical examination of factors influencing the intention to use mobile payment," Computers in Human Behavior, vol. 26, no. 3, 2010, pp. 310–322.

[14] T.-P. Liang and Y.-H. Yeh, "Effect of use contexts on the continuous use of mobile services: the case of mobile games," Personal and Ubiquitous Computing, vol. 15, no. 2, 2011, pp. 187–196.

[15] R. Ali, F. Dalpiaz, and P. Giorgini, "A goal-based framework for contextual requirements modeling and analysis," Requirements Engineering, vol. 15, no. 4, 2010, pp. 439–458.

[16] J. Krogstie, "Requirement engineering for mobile information systems," in Proceedings of the seventh international workshop on requirements engineering: Foundations for software quality (REFSQ01), 2001.

[17] E. Eshet and H. Bouwman, "Addressing the context of use in mobile computing: a survey on the state of the practice," Interacting with Computers, 2014, p. iwu002.

[18] M. Kakihara and C. Sorensen, "Mobility: An extended perspective," in System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on. IEEE, 2002, pp. 1756–1766.

[19] H. B.-L. Duh, G. C. Tan, and V. H.-h. Chen, "Usability evaluation for mobile device: a comparison of laboratory and field tests," in Proceedings of the 8th conference on Human-computer interaction with mobile devices and services. ACM, 2006, pp. 181–186.

[20] S. Kristoffersen and F. Ljungberg, "Mobile use of it," in the Proceedings of IRIS22, Jyväskylä, Finland. Citeseer, 1999.

[21] J. Kjeldskov and J. Stage, "New techniques for usability evaluation of mobile systems," International journal of human-computer studies, vol. 60, no. 5, 2004, pp. 599–620.

[22] B. N. Schilit and M. M. Theimer, "Disseminating active map information to mobile hosts," Network, IEEE, vol. 8, no. 5, 1994, pp. 22–32.

[23] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on. IEEE, 1994, pp. 85–90.

[24] P. J. Brown, "The stick-e document: a framework for creating context-aware applications," Electronic publishing-chichester-, vol. 8, 1995, pp. 259–272.

[25] J. Pascoe, "Adding generic contextual capabilities to wearable computers," in Wearable Computers, 1998. Digest of Papers. Second International Symposium on. IEEE, 1998, pp. 92–99.

[26] T. Rodden, K. Cheverst, K. Davies, and A. Dix, "Exploiting context in hci design for mobile systems," in Workshop on human computer interaction with mobile devices, 1998, pp. 21–22.

[27] G. Chen, D. Kotz et al., "A survey of context-aware mobile computing research," Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, Tech. Rep., 2000.

[28] R. Harrison, D. Flood, and D. Duce, "Usability of mobile applications: literature review and rationale for a new usability model," Journal of Interaction Science, vol. 1, no. 1, 2013, pp. 1–16.

[29] A. Mavin, P. Wilkinson, A. Harwood, and M. Novak, "Easy approach to requirements syntax (ears)," in Requirements Engineering Conference, 2009. RE'09. 17th IEEE International. IEEE, 2009, pp. 317–322.

[30] M. Böhmer, B. Hecht, J. Schöning, A. Krüger, and G. Bauer, "Falling asleep with angry birds, facebook and kindle: a large scale study on mobile application usage," in Proceedings of the 13th international conference on Human computer interaction with mobile devices and services. ACM, 2011, pp. 47–56.

[31] P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimbigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum, and A. L. Wolf, "An architecture-based approach to self-adaptive software," IEEE Intelligent systems, no. 3, 1999, pp. 54–62.

[32] I. D. Constantiou, J. Damsgaard, and L. Knutsen, "The four incremental steps toward advanced mobile service adoption," Communications of the ACM, vol. 50, no. 6, 2007, pp. 51–55.

[33] Z. Zhang and X. Zheng, "User profiles and user requirements in mobile services," Perspectives in Business Information Research-BIR'2007, 2007, p. 170.

[34] Y. Liu and Z. Zhang, "Stakeholder-centered requirements elicitation: A view from user research," in 7th International Conference on Perspectives in Business Information Research, 2008, pp. 25–26.

[35] A. Van Lamsweerde, "Goal-oriented requirements engineering: A guided tour," in Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on. IEEE, 2001, pp. 249–262.

[36] K. W. Thomas and B. A. Velthouse, "Cognitive elements of empowerment: An interpretive model of intrinsic task motivation," Academy of management review, vol. 15, no. 4, 1990, pp. 666–681.

[37] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," Advances in psychology, vol. 52, 1988, pp. 139–183.

[38] S. R. Barley, "On technology, time, and social order: Technically induced change in the temporal organization of radiological work," Making time: Ethnographies of high-technology organizations, 1988, pp. 123–169.

[39] S. Jumisko-Pyykkö and T. Vainio, "Framing the context of use for mobile hci," International Journal of Mobile Human Computer Interaction, vol. 2, no. 4, 2010, pp. 1–28.

[40] L. Chittaro, "Distinctive aspects of mobile interaction and their implications for the design of multimodal interfaces," Journal on Multimodal User Interfaces, vol. 3, no. 3, 2010, pp. 157–165.

[41] K. Schmidt and C. Simonee, "Coordination mechanisms: Towards a conceptual foundation of cscw systems design," Computer Supported Cooperative Work (CSCW), vol. 5, no. 2-3, 1996, pp. 155–200.

[42] F. Ljungberg and C. Sørensen, "Overload: From transaction to interaction," Planet Internet, 2000, pp. 113–136.

[43] Tencent Technology (Shenzhen) Company Limited, "Wechat." [Online]. Available: http://www.wechat.com/

[44] Google, Inc., "Gmail-email from google." [Online]. Available: https://itunes.apple.com/us/app/id422689480

[45] REDDIT, INC., "Alienblue-reddit official client." [Online]. Available: https://itunes.apple.com/us/app/id923187241

 ISBN: 978-1-61208-438-1 

# PUBLICATION

# II

**Sentiment-aware Analysis of Mobile Apps User Reviews Regarding Particular Updates**

X. Li, Z. Zhang and K. Stefanidis

*International Conference on Software Engineering Advances (ICSEA)*, 2018, 109

# Sentiment-aware Analysis of Mobile Apps User Reviews Regarding Particular Updates

Xiaozhou Li, Zheying Zhang, Kostas Stefanidis

Faculty of Natural Sciences, University of Tampere
Tampere, Finland
Email: xiaozhou.li@uta.fi, zheying.zhang@uta.fi, kostas.stefanidis@uta.fi

***Abstract*—The contemporary online mobile application (app) market enables users to review the apps they use. These reviews are important assets reflecting the users needs and complaints regarding the particular apps, covering multiple aspects of the mobile apps quality. By investigating the content of such reviews, the app developers can acquire useful information guiding the future maintenance and evolution work. Furthermore, together with the updates of an app, the users reviews deliver particular complaints and praises regarding the particular updates. Despite that previous studies on opinion mining in mobile app reviews have provided various approaches in eliciting such critical information, limited studies focus on eliciting the user opinions regarding a particular mobile app update, or the impact the update imposes. Hence, this study proposes a systematic analysis method to elicit user opinions regarding a particular mobile app update by detecting the similar topics before and after this update, and validates this method via an experiment on an existing mobile app.**

***Keywords–Mobile app; review; sentiment analysis; topic modeling; topic similarity***

## I. INTRODUCTION

The increasing number of smart phone users has led to a continuous increase in the number of mobile apps and their overall usage. Users browse and download apps via different digital distribution platforms (e.g., Apple app store, and Google Play). These platforms also provide an important channel enabling the users to provide feedback to the app. The ratings and comments given by the users at a particular time reflect their opinions regarding the overall app and the specific version of that app. While the app developers, continuously or sporadically, update their apps, the retrieved user reviews reflect not only their overall opinion changes throughout the evolution time but also their specific complaints and praises regarding the specific app version [1]. Such specific complaints, regarding various aspects [2], shall enable the developers to be aware of the issues and tackle them accordingly.

Existing studies have proposed different approaches to identifying change requests from user reviews for mobile app maintenance. With various opinion mining techniques, such as Natural Language Processing (NLP), Sentiment Analysis (SA) and supervised learning, many studies have been conducted regarding the classification of reviews towards different issue perspectives [3] [4]. Other perspectives, such as user preferences, app evaluation, user satisfaction, relation between download and rating, feature extraction, review prioritization and so on, have also been widely studied [5]–[9]. However, limited studies focus on the use of such methods in opinion mining on particular updates of a mobile app and the impact on the app's updates in the following releases, despite the importance of such information. It is unclear how users' attitude towards a particular issue changes when new updates are released and how the reviews have impacts on app's maintenance and evolution.

In this paper, we investigate the correlation between users' positive and negative reviews before and after an app's release. We consider two dimensions: time and sentiment. Specifically, we divide user reviews based on major updates, and distinguish them between those precede and follow the particular updates. Each group of reviews are further divided into positive and negative ones using sentiment analysis. We devise an approach to measuring the similarity of each group of reviews. The measurement reveals the similarity and changes between different groups of reviews and helps to gain insight into how to detect users' opinion changes regarding a particular update. Furthermore, detecting the users' update-specific opinions shall also help the developers be aware of the users opinion on a particular release and guides them proactively to address the most important issues early.

The remainder of this paper is organized as follows. Section II introduces the method with details. Section III presents a case study using this method. Section IV introduces the related works when Section V concludes the paper.

## II. METHOD DESCRIPTION

In this section, we introduce our method with the main goal to detect the correlation between the content of app reviews before and after the app updates done by the app company through the app maintenance lifecycle. Such correlation shall show the degree in which the users comments are reflected in the sequence of updates and such updates are accepted by the users. The factors that influence and reflect such correlation include the main topics of the reviews between each two updates, the sentiment of those reviews, and the topics similarities before and after each update. Next, we illustrate how to detect the correlation via investigating these factors. Accordingly, Subsection A introduces the overall procedure of the method and brings forth the hypotheses it aims to verify. Subsection B and C introduce respectively how to analyze the sentiment and topics of user reviews. Subsection D introduces how to calculate the similarity between topics, when Subsection E presents how to identify the matching similar topics between review sets.

### A. Preliminaries

Let $R$ be a collection of user reviews for a particular mobile app $A$, covering a particular time period. Therein, each review $r_i \in R$ is associated with a particular time point, at which

 ISBN: 978-1-61208-668-2 

the review $r_i$ is published. Let also $U$ be the set of updates released by the app developers within the time period with each update $u_i$ is released at a particular time point as well. Therefore, we consider each review $r_i$, which is published after the release time of update $u_i$ and before that of the next update $u_{i+1}$, as a review regarding the update $u_i$. Hence, for the $n$ updates $\{u_i | i \in N, k \leq i < k+n\}$ where $k \geq 1$ and $n > 0$ for the app $A$ within a time period, the review set $R$ can be divided into $n+1$ subset where each $R_i \subseteq R$ is the set of reviews commenting on the according $u_i$. $R_0$ is the review set correlated with the last update before $u_1$ or the first version of A if $u_1$ is the first update of A. For each review $r_j \in R_i$, a sentiment score shall be calculated and assigned to $r_j$, whose the sentiment is either $positive$ or $negative$. In this way, by identifying the sentiment of each individual review in $R_i$, we can divide $R_i$ positive review set $R_i^+$ and negative review set $R_i^+$, where $R_i^+ \cup R_i^- = R_i$ and $R_i^+ \cap R_i^- = \emptyset$ with an acceptable accuracy.

We further investigate the main topics for each of the positive and negative review sentence sets. We assign $T_i^+$ and $T_i^-$ as the topic set for the positive and negative reviews. Therefore, we investigate the merits and issues of a particular update $u_i$ by comparing the similarities and changes between $T_{i-1}^+$, $T_{i-1}^-$, $T_i^+$, and $T_i^-$. Specifically, the following hypotheses shall be verified.

- H1. The topic similarities between $T_{i-1}^+$ and $T_i^+$ reflect the merits regarding the app $A$ in general.
- H2. The topic similarities between $T_{i-1}^+$ and $T_i^-$ reflect the uncomfortable changes in the update $u_i$.
- H3. The topic similarities between $T_{i-1}^-$ and $T_i^+$ reflect the improvement in the update $u_i$.
- H4. The topic similarities between $T_{i-1}^-$ and $T_i^-$ reflect the remaining issues regarding the app $A$.

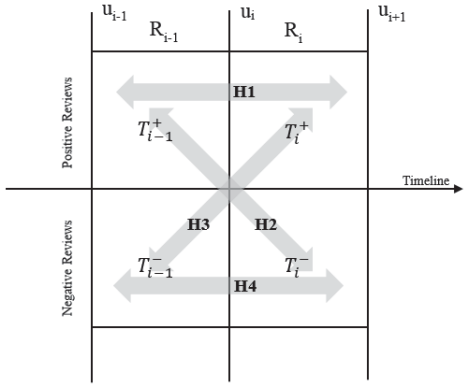


Figure 1. Relationship between hypotheses and updates.

Thus, the results obtained from these hypotheses for updating $u_i$ provide the following information: 1) the merits and issues for the app $A$ in general, 2) the merit and issues specific to the update $u_i$, and 3) the improvements and drawbacks of $u_i$ compared with $u_{i-1}$. The merits and issues for app $A$ and those for each $u_i \in U$ shall be recorded as the evolution status of app $A$ throughout period $T$, which can be used as the reference to guide planning the following updates. Figure 1 visually depicts the focus of our hypotheses for updating $u_i$.

### B. Sentiment Classification

The aim of sentiment classification in this method is to classify each review set $R_i$ into two subsets, i.e., $R_i^+$ and $R_i^-$. Herein, $R_i^+$ denotes the set of positive reviews from $R_i$, and $R_i^-$ denotes the set of negative reviews. Therefore, each $r_j$ in $R_i$ shall be determined whether it is positive or negative.

To do so, we assign a sentiment score to each review by exploiting a robust tool for sentiment strength detection on social web data [10]. As each $r_j$ can be seen as a list of words $W_j$, we first select a lexicon that will determine the sentiment score of each word $w_z$ in $W_j$. The lexicon for sentiment analysis is a list of words used in English language, each of which is assigned with a sentiment value in terms of its sentiment valence (intensity) and polarity (positive/negative). To determine the sentiment of words, we assign a rational value within a range to a word. For example, if the word "okay" has a positive valence value of 0.9, the word "good" must have a higher positive value, e.g., 1.9, and the word "great" has even higher value, e.g., 3.1. Furthermore, the lexicon set shall include social media terms, such as Western-style emoticons (e.g., :-)), sentiment-related acronyms and initialisms (e.g., LOL, WTF), and commonly used slang with sentiment value (e.g., nah, meh).

**Algorithm 1** Sentiment Classification

```
1: procedure SENTIMENT ANALYSIS
2: lexicons ← selected sentiment lexicon
3: R_i ← review set
4: W_j ← word set of r_j(r_j ∈ R_i)
5: For each r_i in R_i
6:     For each w_z in W_j
7:         s_z ← polarity and valence analysis(w_z, lexicons)
8:         list ← list.append(s_z)
9:     S_j ← grammatical and syntactical heuristics
       (r_j, list, heuristics)
10:    if S_j > 0 then
11:        R_i^+.append(r_j).
12:    if S_j = 0 then
13:        R_i^0.append(r_j).
14:    if S_j < 0 then
15:        R_i^-.append(r_j).
16: trainMatrix ← train(R_i^+[: 8000], R_i^-[: 8000])
17: R_i^0+, R_i^0- ← naive bayes classifier(R_i^0, trainMatrix)
18: R_i^+.extend(R_i^0+)
19: R_i^-.extend(R)i^0-)
20: return R_i^+, R_i^-
```

Figure 2. Algorithm for Sentiment Classification

With the well-established lexicon, and a selected set of proper grammatical and syntactical heuristics, we shall then be able to determine the overall sentiment score of a review. Namely, the sentiment score of a review $r_j$ is equal to $S_j$, where $S_j \in (-1, 1)$. The grammatical and syntactical heuristics are seen as the cues to change the sentiment of word sets. Therein, punctuation, capitalization, degree modifier, and contrastive conjunctions are all taken into account. For example, the sentiment of "The book is EXTREMELY AWESOME!!!" is stronger than "The book is extremely awesome", which is

 ISBN: 978-1-61208-668-2 

stronger than "The book is very good.". With both the lexicon value for each word of the review, and the calculation based on the grammatical and syntactical heuristics, we can then assign unique sentiment values to each review. That is, each review $r_j$ is classified into positive, neutral or negative, as following:

$$r_j \text{ is } \begin{cases} \text{positive, if } 0 < S_j < 1, \\ \text{neutral, if } S_j = 0, \\ \text{negative, if } -1 < S_j < 0. \end{cases}$$

Overall, each review set $R_i$ is divided into $R_i^+$, $R_i^0$, and $R_i^-$, denoting the positive, neutral and negative review sets. To further investigate the information in $R_i^0$, after experimentally observing that typically includes a big number of reviews, we classify it into positive and negative using the Naive Bayes Classifier with the training data from $R_i^+$ and $R_i^-$. This way, $R_i^0$ is classified into $R_i^{0+}$ and $R_i^{0-}$, which in turn, are added to $R_i^+$ and $R_i^-$, respectively. The reason to perform supervised classification after sentiment analysis instead of directly applying classification is twofold. Firstly, manually creating training data is time-consuming and less accurate than using existing sentiment analysis methods. Secondly, training the sentiment classified reviews will provide domain specific and reliable results. The process is described in Figure 2.

### C. Topic Analysis

After dividing the review sets $R_i$ and $R_{i-1}$, i.e., the review sets related to update $u_i$, into $R_i^+$, $R_i^-$, $R_{i-1}^+$ and $R_{i-1}^-$ based on the sentiment classification method, we elicit the main topics from each of the classified review sets by exploiting the Latent Dirichlet Allocation (LDA) method [11]. First, we consider each review sentence $r_j$ in a particular set of reviews as a list of words $W_j$, where the sequence of the words is not recorded. The number of topics in this review set is set as $t$. Presumably, there is a distribution for the probability of a particular word appears in a particular topic, when there is also one for that of a particular review in a topic. We build the set of $Review - Topic$, where each word of each review is assigned with a topic out of the $t$ topics. As preparation, we define $Review{-}topic{-}numbers$, $Topic{-}words{-}numbers$, and $Topic - numbers$ denoting the number of occurrence of each topic in each review, the number of occurrence of each word in each topic, and the number of words in each topic, respectively. For example, $Review - topic - numbers(r_j, k)$ denotes the number of occurrences of topic $k$ in review $r_j$. Then, we randomly assign each word $w_z$ of each review $r_j$ with a topic $t_k$. Accordingly, the $Review - topic - numbers$, $Topic - words - numbers$, and $Topic - numbers$ will be updated as the referencing weight of the distribution the words for each topic. Then iteratively, for each word, we assign a new topic based on such weight of distribution and adjust the weight with the $Review - topic - numbers$, $Topic - words - numbers$, and $Topic - numbers$ for the next iteration. After a given number of iteration, $t$ topics will be determined by the $Topic - words - numbers$, which is the number of occurrences of the words in each topic. Each $t_k$ is then denoted by the most common keywords used in this topic. Then, the set of topics $T$ are returned as result. For the review sets $R_i^+$, $R_i^-$, $R_{i-1}^+$ and $R_{i-1}^-$, we will have the topics sets $T_i^+$, $T_i^-$, $T_{i-1}^+$ and $T_{i-1}^-$ accordingly.

### D. Calculating Topics Similarities

Based on the topic sets $T_i^+$, $T_i^-$, $T_{i-1}^+$ and $T_{i-1}^-$ elicited from the review sets $R_i^+$, $R_i^-$, $R_{i-1}^+$ and $R_{i-1}^-$, we further analyze the similarities between the individual topics between each pair of the topic sets. As the result from the previous topic analysis, each topic set $T$ encompasses $k$ topics, each of which is represented by the list of the most possible appearing keywords. Thus, each topic set $T$ with $k$ topics each of which is represented by $w$ keywords, can be denoted as:

$$T = \begin{bmatrix} kw_{1,1} & kw_{1,2} & ... & kw_{1,w} \\ ... & ... & ... & ... \\ kw_{k,1} & kw_{k,2} & ... & kw_{k,w} \end{bmatrix}$$

with each $t_i \in T$ can be denoted as $[kw_{i,1}, kw_{i,2}, ...kw_{i,k}]$. To compare the similarity between two topic sets, each consisting of $t$ topics, we compare all pairs of topics. Due to the fact that each topic is represented as a set of keywords, the similarity of two topics shall be denoted by the common keywords of these topics. Hence, an easy way for calculating the similarity between any two topics $t_i$ and $t_j$ is by using the Jaccard similarity. This similarity function reflects the percentage of the common keywords of the two sets in the whole keywords set of the two: $J(t_i, t_j) = \frac{|t_i \cap t_j|}{|t_i \cup t_j|}$.

However, by using the Jaccard Similarity, we consider two given topics are similar only when they contain a particular number of common keywords, regardless of the probability of them. The meaning of each topic $t_i \in T$, denoted as $[kw_{i,1}, kw_{i.2}, ...kw_{i,k}]$, shall be more likely reflected by the high-probability keywords of $t_i$. Furthermore, the subset of only low-probability keywords may reflect different meanings. For example, a topic is denoted as {'update': 0.143, 'problem': 0.096, 'fix': 0.064, 'install': 0.03, 'uninstall': 0.029, 'open': 0.027, 'stop': 0.025, 'plea': 0.025, 'get': 0.022, 'reinstall': 0.019, 'bug': 0.019, 'start': 0.019, 'need': 0.014, 'application': 0.014, 'issue': 0.011, 'battery': 0.011, 'help': 0.01, 'face': 0.009, 'frustrate': 0.009, 'day': 0.008}. From the high-probability keywords of this topic, we can summarize that the topic is regarding the problems of updating, which requires being fixed. However, the low-probability keywords hardly reflect the topic, e.g., a keyword subset, {'reinstall', 'bug', 'start', 'need', 'application', 'issue', 'battery'}, reflects a very different issue regarding bugs and batteries.

Hence, when comparing the similarity of two given topics, the probability of the common keywords shall be taken into account. Considering that Jaccard coefficient is the normalized inner product [12], we herein adopt the similarity measure method incorporating also the inner product, the Kumar-Hassebrook (KH) similarity [13]. Provided between topic $t_i$ and $t_j$, the $c$ common keywords are denoted as $[kw_{ij,1}, kw_{ij,2}, ...kw_{ij,c}]$, with the according probability list in $t_i$ and $t_j$ is $[p_{i,1}, p_{i,2}, ...p_{i,c}]$ and $[p_{j,1}, p_{j,2}, ...p_{j,c}]$. The similarity of the two given topics are calculated as follows.

$$KH(t_i, t_j) = \frac{\sum_{x=1}^{c} p_{i,x} \cdot p_{j,x}}{\sum_{x=1}^{k} p_{i,x}^2 + \sum_{x=1}^{k} p_{j,x}^2 - \sum_{x=1}^{c} p_{i,x} \cdot p_{j,x}}$$

The probability for each keyword of any topic belongs to (0,1). Hence, for this formula, when $t_i$ and $t_j$ contain more common keywords, the numerator increases monotonically, and the denominator decreases monotonically. Therefore, $KH(t_i, t_j)$ increases when $t_i$ and $t_j$ have more keywords in

 ISBN: 978-1-61208-668-2

common. In addition, when the probability of the common keywords increases, $\sum_{x=1}^{c} p_{i,x} \cdot p_{j,x}$ increases. Because the denominator is greater than the numerator, and both are greater than 0, $KH(t_i, t_j)$ increases when the probabilities of the common keywords of $t_i$ and $t_j$ increase.

In this way, for two given topics $t_i$ and $t_j$, when each keyword of these two topics is assigned the average value of the probability value set, then $KH(t_i, t_j) = J(t_i, t_j)$. Considering the monotonical increasing of the KH Similarity formula, it means that for $t_i$ and $t_j$, when $KH(t_i, t_j) < J(t_i, t_j)$, the common keywords of these two topics hardly reflect the meaning of them. For example, two topics, denoted as the following set of keywords with the according probability of each keywords, are listed in Table I.

TABLE I. EXAMPLE TOPICS WITH KEYWORDS AND PROBABILITY

| | |
|---|---|
| Topic 1 | {'problem': 0.145, 'fix': 0.081, 'download': 0.051, 'please': 0.032, 'use': 0.025, 'reason': 0.021, 'service': 0.02, 'user': 0.015, 'issue': 0.015, 'data': 0.014} |
| Topic 2 | {'version': 0.197, 'please': 0.121, 'go': 0.069, 'use': 0.043, 'option': 0.036, 'one': 0.028, 'lot': 0.027, 'revert': 0.021, 'way': 0.018, 'download': 0.018} |

The Jaccard Similarity of these two topic is 0.176 when the KH Similarity is 0.064. We can observe that the common keywords, {'download', 'please', 'use'}, are of low probability in Topic 1 and keywords {'please', 'use'} in Topic 2, while neither topic is reflected by the common keywords. Topic 1 can be seen regarding the requests of fixing problems/bugs when Topic 2 is more related to keyword 'version' instead of 'download'. Thus, these two given topics cannot be considered as similar despite the high Jaccard Similarity.

### E. Identifying Matching Topics

After computing similarities between pairs of review topics with KH similarity, we shall identify which are the matching topics when cross-comparing the topics of the topics sets $T_i^+$, $T_i^-$, $T_{i-1}^+$ and $T_{i-1}^-$. Hence, to identify the matching topics between two review topic sets $T_a$ and $T_b$, the aim is to identify all the topic pairs ($t_{ai}$, $t_{bj}$), $t_{ai} \in T_a$ and $t_{bj} \in T_b$, that have the high similarity. Starting from the pair of topics with highest similarity values,

We firstly use the Jaccard Similarity value of two particular topics as the threshold for their KH similarity. According to the formula of KH similarity given previous, we set the probability of each keyword in each topic equal to the average. Then

$$KH(t_{ai}, t_{bj}) = \frac{\sum_{x=1}^{c} p^2}{\sum_{x=1}^{k} p^2 + \sum_{x=1}^{k} p^2 - \sum_{x=1}^{c} p^2}$$

$$= \frac{c}{k + k - c} = J(t_{ai}, t_{bj})$$

Therefore, we select the topic pairs, whose KH similarity value greater than their Jaccard similarity value, as similar topics. Furthermore, from the topic pairs with the highest KH similarity value, we select the top $n$ pairs of topics to investigate the changes and similarities of users opinion regarding the app.

**Algorithm 2** Matching Topics Identification

```
procedure MATCHING TOPICS IDENTIFICATION
k ← number of topics
n ← number of selected similar topics
Ta ← {t_a1, t_a2, ..., t_ai, ..., t_ak}
Tb ← {t_b1, t_b2, ..., t_bj, ..., t_bk}
KHs_ij ← KHSimilarity(t_ai, t_bj)
Js_ij ← JaccardSimilarity(t_ai, t_bj)
For i, j ∈ {1, 2, ..., k}
    if KHs_ij ≥ Js_ij then
        S ← s_ij
If len(S) > n
    For each (t_ax, t_by) in S[:n]:
    Result.append(Summarize(t_ax, t_by))
Elsc:
    For each (t_ax, t_by) in S:
    Result.append(Summarize(t_ax, t_by))
Return Result
```

Figure 3. Algorithm for Matching Topic Identification

The aim of the algorithm (shown in Figure 3) is to select the similar topic pairs with the highest similarity value. When a particular topic pair is selected, the other pairs, which either of these two selected topics is also pairing with and have also high similarity, will be considered as references to interpret the users opinions. Each particular topic generated by the LDA model contains a number of perspectives that can be interpreted by the keywords. Thus, it is possible that one particular topic have the similar similarity value to multiple topics, when they are similar regarding different perspectives which represented by their different common keywords.

TABLE II. EXAMPLE TOPICS WITH KEYWORDS AND PROBABILITY

| | |
|---|---|
| Topic 1 | {'time': 0.12, 'friend': 0.091, 'talk': 0.081, 'way': 0.069, 'see': 0.048, 'people': 0.045, 'communication': 0.038, 'want': 0.03, 'face': 0.023, 'world': 0.02 } |
| Topic 2 | {'friend': 0.111, 'bring': 0.07, 'connect': 0.068, 'talk': 0.06, 'keep': 0.059, 'family': 0.054, 'application': 0.037, 'way': 0.021, 'touch': 0.017, 'contact': 0.016 } |
| Topic 3 | {'see': 0.077, 'use': 0.056, 'people': 0.046, 'want': 0.044, 'contact': 0.034, 'thing': 0.031, 'year': 0.027, 'find': 0.023, 'know': 0.023, 'number': 0.019 } |

For example, in Table II Topic 1 has the same Jaccard similarity value to both Topic 2 and 3. The two pairs of common keywords are {'friend', 'talk', 'way'} and {'people', 'see', 'want'}. Their KH similarity values are different but both high (0.276 and 0.137). From Topic 1, we could summarize that it is regarding using the app enabling people to communicate with friends any time they want and can see their faces as well. Despite it is considered similar to both Topic 2 and 3, Topic 2 focuses on the perspective of enabling communication between families and friends, when Topic 3 focus more on the perspective of contacting people with phone numbers and seeing them. Thus, by identifying both similar topic pairs, we shall have more thorough understanding of the users' opinions regarding the app.

## III. CASE STUDY

### A. Preprocessing

Before starting the experiment with the proposed method, preprocessing on the raw review data is required. The whole

 ISBN: 978-1-61208-668-2 

preprocessing work can be divided into three individual steps as follows.

**Filtering non-English reviews.** The raw review data may contain a number of review items that are not written in English, which needs to be filtered out. Also, similar to social media text, user reviews usually contain many commonly used slurs that are not regular English vocabularies. Our goal is to not filter out these words, as they likely contain sentiment related information, without which shall influence our experiment results. Overall, we screen out the non-English review sentences using Langdetect [14], a convenient language detecting package for Python language. Compared with PyEnchant [15], another language detecting package, Langdetect enables determining the language of text on sentence level. It shall remain the review data containing such English slurs.

**Focusing on sentence-level granularity.** Due to the fact that each user review can contain more than one sentence, a multi-sentence review can contain multiple meanings, one for each sentence. Thus, we divide each review from a review set $R_i$ into individual sentences. Hereafter, we use $r_j$ to denote a review sentence in $R_i$. We use the sentence tokenizer feature from the NLTK [16] python package, with a further checking on the legitimacy of the sentences.

**Filtering stop-words and lemmatization.** In addition, the collected English review sentences are also transformed into lower cases, screened with stop-words, and lemmatized before topic modeling. In order to obtain more meaningful topic modeling results, we add the words that connect to only general information but have significant appearing rate in the reviews to the list of stop words. For example, the name of the app and the word app are of neither help towards topic modeling nor towards sentiment analysis. In addition, considering the fact that the sentiment of each review item is identified, we eliminate all adjectives from the obtained tokens and select only nouns and verbs as tokens via the pos_tag function of NLTK.

**Sentiment Classification with VADER.** To perform sentiment analysis on the collected app reviews, we select the Valence Aware Dictionary for sEntiment Reasoning (VADER) approach [10]. Compared with other sentiment analysis tools, VADER has a number of advantages regarding this study. Firstly, the classification accuracy of VADER on sentiment towards positive, negative and neutral classes is even higher than individual human raters in social media domain. In addition, its overall classification accuracies on product reviews from Amazon, movie reviews, and editorials from NYTimes also outperform other sentiment analysis approaches, such as SenticNet [17], SentiWordNet [18], Affective Norms for English Words [19], and Word-Sense Disambiguation [20], and run closely with the accuracy of individual human. On the other hand, VADER approach is integrated in the NLTK package, which can be easily imported and performed using Python.

### B. Dataset

Our study relies on real data. In particular, we focus on reviews submitted for 1-year period of Skype on the Android platform. We collected 153,128 user reviews submitted from 1.9.2016 to 31.8.2017. The reviews are tokenized into 234,064 individual sentences. After filtering the non-English review sentences, the number is reduced to 174,559.

We investigate the merits and issues concerning the major update of Skype released on Android platform on 1.6.2017 ($u_i$ = version-1.6.2017). Within this period from 1.9.2016 to 31.8.2017, 76 updates were released. On average, the app has been updated nearly every five days. By observing the content of each update from the given information of Google Play, we find that some consecutive updates contain exact same content based on their descriptions. Therefore, we consider the first update of a set of updates which contain same descriptions as a major update, when the rest of the update set as minor update. Amongst the major updates of Skype during this period, the update $u_i$ provide significant changes in UI design and user experiences. By classifying all selected review sentences into positive and negative using sentiment analysis and supervised classification with Naive Bayes Classifier, the number of review sentences in each segment is listed in Table III. Accordingly, the review sets $R^+_{i-1}$, $R^-_{i-1}$, $R^+_i$, and $R^-_i$, for this study, consists of 65580, 29970, 36703, and 42306 reviews.

TABLE III. POSITIVE AND NEGATIVE REVIEWS AROUND A PARTICULAR APP UPDATE

| | Positive reviews | Negative reviews | Total |
|---|---|---|---|
| Before 1.6.2017 | 65,580 | 29,970 | 95,550 |
| After 1.6.2017 | 36,703 | 42,306 | 79,009 |
| Total | 102,283 | 72,276 | 174,559 |

Overall, the total number of positive reviews around the particular update is bigger than the number of negative reviews. Meanwhile, the monthly review number increased sharply after this particular major update. Opposite to the situation before the update, we observe that after the update more negative reviews are given by the users than the positive ones, meaning that many users are not satisfied with this particular update or the app overall.

TABLE IV. NUMBER OF REVIEWS PER TOPIC

| $t^+_{(i-1)1}$ | $t^+_{(i-1)2}$ | $t^+_{(i-1)3}$ | $t^+_{(i-1)4}$ | $t^+_{(i-1)5}$ |
|---|---|---|---|---|
| 13627 | 3104 | 3168 | 6283 | 4725 |
| $t^-_{(i-1)1}$ | $t^-_{(i-1)2}$ | $t^-_{(i-1)3}$ | $t^-_{(i-1)4}$ | $t^-_{(i-1)5}$ |
| 8692 | 6016 | 4122 | 6105 | 9738 |
| $t^+_{i1}$ | $t^+_{i2}$ | $t^+_{i3}$ | $t^+_{i4}$ | $t^+_{i5}$ |
| 3519 | 2892 | 4204 | 1895 | 3433 |
| $t^-_{i1}$ | $t^-_{i2}$ | $t^-_{i3}$ | $t^-_{i4}$ | $t^-_{i5}$ |
| 2409 | 2700 | 2794 | 2408 | 3716 |
| $t^+_{(i-1)6}$ | $t^+_{(i-1)7}$ | $t^+_{(i-1)8}$ | $t^+_{(i-1)9}$ | $t^+_{(i-1)10}$ |
| 7768 | 2796 | 3134 | 3186 | 3391 |
| $t^-_{(i-1)6}$ | $t^-_{(i-1)7}$ | $t^-_{(i-1)8}$ | $t^-_{(i-1)9}$ | $t^-_{(i-1)10}$ |
| 4810 | 3399 | 3244 | 2798 | 2177 |
| $t^+_{i6}$ | $t^+_{i7}$ | $t^+_{i8}$ | $t^+_{i9}$ | $t^+_{i10}$ |
| 4818 | 4132 | 4463 | 3237 | 4635 |
| $t^-_{i6}$ | $t^-_{i7}$ | $t^-_{i8}$ | $t^-_{i9}$ | $t^-_{i10}$ |
| 3139 | 3893 | 3229 | 6252 | 4508 |

After sentiment analysis and classification, we perform the LDA topic analysis to identify the topics of the review set $R^+_{i-1}$, $R^-_{i-1}$, $R^+_i$, and $R^-_i$, in order to investigate the users opinions concerning the update and further verify the previously proposed hypothesis. To train the LDA topic models, we need to set the number of topics $k$. Based on an experimentally study regarding the quality of the topics produced for different $k$ values, and select $k = 10$.

Overall, for the collected 174,559 review sentences on Skype, we perform an LDA topic analysis on the review set $R^+_{i-1}$, $R^-_{i-1}$, $R^+_i$, and $R^-_i$. For each of the 4 sets of review data,

 ISBN: 978-1-61208-668-2

we use the Gensim topic modeling toolbox to train the 10-topic LDA models. For the 4 LDA models, each individual topic is represented by 20 keywords. Then, we assign each review sentence in the LDA models to the topic to which it has the highest probability to belong. The numbers of reviews for each topic of the 4 data sets appear in Table IV ($t^+_{in}$ represents the $n^{th}$ topic of $R^+_i$). In general, reviews are divided into topics smoothly and, in most cases, each topic consists of around 4000 review sentences.

### C. Topics Similarity Analysis

With the 40 identified topics, each of which is represented by 20 keywords, we compare the similarity between each topic from the 20 topics before the update and each one from the 20 topics after the update, which provides 400 topics pairs. The Jaccard similarity values for those 400 pairs range from 0 to 0.429 (i.e., 0 - 12 common keywords between two topics). Meanwhile, the KH similarity values range from 0 to 0.676. Therein, we identify the potential similar topics from the highest KH similarity value and eliminate the results where the KH similarity value is lower than the according Jaccard similarity value. In this way, we guarantee the common keywords of each identified topic pairs contain the overall probability value greater than average. We select the seven topic pairs with the highest KH similarity values for each comparison set. For each topic pair, we analyze the similarity of the two topics by observing their common keywords. Furthermore, from the unique keywords of each topic from one particular topic pair, we could also see the topic changes. The seven topic pairs with the highest KH similarity value are depicted in Table V.

TABLE V. PAIRS OF SIMILAR TOPICS

| | |
|---|---|
| $T^+_{i-1}$ - $T^+_i$ | $(t^+_{(i-1)8}, t^+_{i5})(t^+_{(i-1)6}, t^+_{i6})(t^+_{(i-1)7}, t^+_{i9})(t^+_{(i-1)6}, t^+_{i1})$ $(t^+_{(i-1)6}, t^+_{i5})(t^+_{(i-1)1}, t^+_{i7})(t^+_{(i-1)8}, t^+_{i3})$ |
| $T^+_{i-1}$ - $T^-_i$ | $(t^+_{(i-1)6}, t^-_{i10})(t^+_{(i-1)6}, t^-_{i5})(t^+_{(i-1)8}, t^-_{i7})(t^+_{(i-1)5}, t^-_{i6})(t^+_{(i-1)7}, t^-_{i8})(t^+_{(i-1)8}, t^+_{i10})(t^+_{(i-1)2}, t^+_{i3})$ |
| $T^-_{i-1}$ - $T^+_i$ | $(t^-_{(i-1)3}, t^+_{i6})(t^-_{(i-1)7}, t^+_{i3})(t^-_{(i-1)5}, t^+_{i1})(t^-_{(i-1)3}, t^+_{i5})$ $(t^-_{(i-1)10}, t^+_{i8})(t^+_{(i-1)1}, t^+_{i4})(t^+_{(i-1)6}, t^+_{i9})$ |
| $T^-_{i-1}$ - $T^-_i$ | $(t^-_{(i-1)3}, t^-_{i10})(t^-_{(i-1)10}, t^-_{i3})(t^-_{(i-1)6}, t^-_{i8})(t^-_{(i-1)7}, t^-_{i9})(t^+_{(i-1)7}, t^+_{i6})(t^+_{(i-1)8}, t^+_{i6})(t^-_{(i-1)8}, t^-_{i9})$ |

By further analyzing the common keywords in the obtained similar topic pairs, we verify the hypothesis H1 - H4 as follows.

**H1. The topic similarities between $T^+_{i-1}$ and $T^+_i$ reflect the merits regarding the app $A$ in general.** The common keywords of the seven topics pairs with the highest KH similarity value appear in Table VI. For example, the common keywords of topics ($t^+_{(i-1)8}$, $t^+_{i5}$) reflects the acknowledgement from the users before and after the update. Because, the common keywords 'work' and 'call' indicate that the core function of the app works properly. The common keywords of topics ($t^+_{(i-1)6}$, $t^+_{i6}$) reflect the users acknowledge also the benefits brought by the app, e.g., enabling people to chat in groups, with video or by calling, so that people can see and hear from their friends. On the other hand, topic pair ($t^+_{(i-1)7}$, $t^+_{i9}$) contains the common keywords that reflect the users needs in adding features and fixing bugs. Considering the overall positive sentiment of the review sets, it is also possible the users reflect their satisfaction towards the work done by the developers regarding such matters. Topic pair ($t^+_{(i-1)6}$, $t^+_{i5}$) reflects the users satisfied with the video and audio quality of the app. As topic $t^+_{i5}$ is also similar to $t^+_{(i-1)8}$, both topic pairs reflect similar information. Topic pair ($t^+_{(i-1)1}$, $t^+_{i7}$) reflects the contribution of the app to the society in a bigger picture, regarding the communication between friends and family and helping people keep in touch. Topic pair ($t^+_{(i-1)6}$, $t^+_{i1}$) contain few common keywords; however, as $t^+_{(i-1)6}$ is also similar to topic $t^+_{i6}$ and $t^+_{i5}$, all these topic pairs reflect similar information.

TABLE VI. COMMON KEYOWRDS IN POSITIVE-POSITIVE REVIEWS

| Topic Pairs | Common Keywords |
|---|---|
| ($t^+_{(i-1)8}$, $t^+_{i5}$) | ['call', 'phone', 'sound', 'work'] |
| ($t^+_{(i-1)6}$, $t^+_{i6}$) | ['call', 'chat', 'friend', 'group', 'hear', 'make', 'people', 'person', 'phone', 'see', 'video'] |
| ($t^+_{(i-1)7}$, $t^+_{i9}$) | ['add', 'bug', 'everything', 'fix', 'hope', 'issue', 'make', 'need', 'please'] |
| ($t^+_{(i-1)6}$, $t^+_{i1}$) | ['people', 'use', 'year'] |
| ($t^+_{(i-1)6}$, $t^+_{i5}$) | ['call', 'make', 'phone', 'quality', 'video', 'voice'] |
| ($t^+_{(i-1)1}$, $t^+_{i7}$) | ['application', 'communicate', 'connect', 'family', 'friend', 'get', 'help', 'touch', 'way'] |
| ($t^+_{(i-1)8}$, $t^+_{i3}$) | ['connection', 'internet', 'keep', 'nothing', 'work'] |

**H2. The topic similarities between $T^+_{i-1}$ and $T^-_i$ reflect the uncomfortable changes in the update $u_i$.** The common keywords of the selected similar topic pairs are shown in Table VII. The common keywords of topics ($t^+_{(i-1)6}$, $t^-_{i10}$) reflects negative user reviews regarding the apps core feature exist, despite the positive feedback before this update. According to $t^-_{i10}$, the aspects which users complain about include calling in general, the user interface, video and sound quality, and connections. Meanwhile, $t^-_{i10}$ is also considered similar to $t^+_{(i-1)8}$. Topic $t^+_{(i-1)8}$ indicates that before the update many users like the internet connection of this app with wifi on computer and tablet. Topic pair ($t^+_{(i-1)5}$, $t^-_{i6}$) reflects that issues concerning the user accounts, including logging in, signing up, passwords emerge after the update, where the users complain quite often. Furthermore, the topic pair ($t^+_{(i-1)7}$, $t^-_{i8}$) indicates that many users complain about the developers fixing problems negatively, despite many others reflect the issue with positive sentiment (see topic pair ($t^+_{(i-1)7}$, $t^+_{i9}$)).

TABLE VII. COMMON KEYOWRDS IN POSITIVE-NEGATIVE REVIEWS

| Topic Pairs | Common Keywords |
|---|---|
| ($t^+_{(i-1)6}$, $t^-_{i10}$) | ['call', 'connect', 'hear', 'make', 'person', 'phone', 'quality', 'video', 'voice'] |
| ($t^+_{(i-1)6}$, $t^-_{i5}$) | ['make', 'use', 'year'] |
| ($t^+_{(i-1)8}$, $t^-_{i7}$) | ['need', 'work'] |
| ($t^+_{(i-1)5}$, $t^-_{i6}$) | ['account', 'go', 'keep', 'let', 'log', 'password', 'sign', 'try', 'win'] |
| ($t^+_{(i-1)7}$, $t^-_{i8}$) | ['fix', 'please', 'problem', 'thing'] |
| ($t^+_{(i-1)8}$, $t^-_{i10}$) | ['call', 'connection', 'drop', 'phone', 'sound', 'work'] |
| ($t^+_{(i-1)2}$, $t^-_{i3}$) | ['conversation', 'get', 'take', 'time', 'type'] |

**H3. The topic similarities between $T^-_{i-1}$ and $T^+_i$ reflect the improvement in the update $u_i$.** The common keywords of the selected similar topic pairs are shown in Table VIII. The common keywords of topics ($t^-_{(i-1)3}$, $t^+_{i6}$) reflect that before the update, many users have complaint regarding using the app for phone calls in general. After the update, a number of positive

 ISBN: 978-1-61208-668-2

reviews towards this matter appear. Considering the topic pair ($t^-_{(i-1)3}$, $t^+_{i5}$), users also switch the attitude regarding the video and sound quality to positive after the update. Topic pair ($t^-_{(i-1)7}$, $t^+_{i3}$) reflects the general positive acknowledgement of the update. Topic pair ($t^-_{(i-1)10}$, $t^+_{i8}$) reflects the users attitude towards the message notification feature has changed into positive. ($t^-_{(i-1)1}$, $t^+_{i4}$) reflects the general positive feedback regarding the new version while ($t^-_{(i-1)6}$, $t^+_{i9}$) reflects the positive feedback on having some bugs fixed in this version.

TABLE VIII. COMMON KEYOWRDS IN NEGATIVE-POSITIVE REVIEWS

| Topic Pairs | Common Keywords |
|---|---|
| ($t^-_{(i-1)3}$, $t^+_{i6}$) | ['call', 'get', 'hear', 'make', 'people', 'person', 'phone', 'see', 'talk', 'time', 'video'] |
| ($t^-_{(i-1)7}$, $t^+_{i3}$) | ['get', 'update', 'win', 'work'] |
| ($t^-_{(i-1)5}$, $t^+_{i1}$) | ['get', 'try', 'use', 'year'] |
| ($t^-_{(i-1)3}$, $t^+_{i5}$) | ['call', 'make', 'phone', 'quality', 'sound', 'time', 'video', 'voice'] |
| ($t^-_{(i-1)10}$, $t^+_{i8}$) | ['message', 'notification', 'open', 'see', 'send', 'show', 'take'] |
| ($t^-_{(i-1)1}$, $t^+_{i4}$) | ['version'] |
| ($t^-_{(i-1)6}$, $t^+_{i9}$) | ['bug', 'fix', 'get', 'lot', 'please'] |

**H4. The topic similarities between $T^-_{i-1}$ and $T^-_i$ reflect the remaining issues regarding the app $A$.** The common keywords of the selected similar topic pairs are shown in Table IX. For example, the common keywords of topics ($t^-_{(i-1)3}$, $t^-_{i10}$) reflects the users complaint regarding the general quality of the app remains after the update, specifically concerning connections, calling, audio and video quality, etc. On the other hand, topic pair ($t^-_{(i-1)10}$, $t^-_{i3}$) reflects the problem with sending and receiving messages with notifications still exist. Furthermore, crashing is also a persisting issue that many users complained about based on topic pair ($t^-_{(i-1)6}$, $t^-_{i8}$). Topic pair ($t^-_{(i-1)7}$, $t^-_{i9}$), despite having only one common keyword, reflects the users general negativity towards the update. Topic pair ($t^-_{(i-1)7}$, $t^-_{i6}$) and ($t^-_{(i-1)8}$, $t^-_{i6}$) reflect the issues regarding logging in and signing up with Microsoft account. Topic pair ($t^-_{(i-1)8}$, $t^-_{i9}$) reflects the issues regarding user contact list when specially $t^-_{i9}$ reflects the users' complaints regarding contact list syncing and status.

TABLE IX. COMMON KEYOWRDS IN NEGATIVE-NEGATIVE REVIEWS.

| Topic Pairs | Common Keywords |
|---|---|
| ($t^-_{(i-1)3}$, $t^-_{i10}$) | ['call', 'connect', 'drop', 'hear', 'make', 'person', 'phone', 'quality', 'sound', 'time', 'video', 'voice'] |
| ($t^-_{(i-1)10}$, $t^-_{i3}$) | ['get', 'message', 'notification', 'open', 'see', 'send', 'show', 'take', 'time'] |
| ($t^-_{(i-1)6}$, $t^-_{i8}$) | ['crash', 'fix', 'give', 'keep', 'please', 'problem', 'star', 'think', 'time'] |
| ($t^-_{(i-1)7}$, $t^-_{i9}$) | ['update'] |
| ($t^-_{(i-1)7}$, $t^-_{i6}$) | ['let', 'login', 'microsoft', 'time', 'try', 'turn', 'update', 'win'] |
| ($t^-_{(i-1)8}$, $t^-_{i6}$) | ['account', 'keep', 'make', 'sign'] |
| ($t^-_{(i-1)8}$, $t^-_{i9}$) | ['add', 'contact', 'list', 'sync'] |

Conclusively, we could summarize the users' opinion before and after the particular update (version 1.6.2017) as follows:

**The merits in general:**

1) ($t^+_{(i-1)8}$, $t^+_{i5}$): calling feature works.
2) ($t^+_{(i-1)6}$, $t^+_{i6}$), ($t^+_{(i-1)1}$, $t^+_{i7}$): people can chat in group with video and calls, connecting with family and friends.
3) ($t^+_{(i-1)7}$, $t^+_{i9}$): added features and bugs fixed.
4) ($t^+_{(i-1)6}$, $t^+_{i5}$): the video and sound quality.

**The uncomfortable changes:**

1) ($t^+_{(i-1)6}$, $t^-_{i10}$): user interface, connection, and calling quality in general.
2) ($t^+_{(i-1)5}$, $t^-_{i6}$): the user accounts, including logging in, signing up, passwords.
3) ($t^+_{(i-1)7}$, $t^-_{i8}$): bugs fixes.

**The improvement:**

1) ($t^-_{(i-1)7}$, $t^+_{i3}$): update in general.
2) ($t^-_{(i-1)10}$, $t^+_{i8}$): message notification.
3) ($t^-_{(i-1)3}$, $t^+_{i6}$), (($t^-_{(i-1)3}$, $t^+_{i5}$): calling in general, video and sound quality.

**The remaining issues:**

1) ($t^-_{(i-1)3}$, $t^-_{i10}$): update in general.
2) ($t^-_{(i-1)10}$, $t^-_{i3}$): sending and receiving messages with notifications.
3) ($t^-_{(i-1)6}$, $t^-_{i8}$): crashing.
4) ($t^-_{(i-1)7}$, $t^-_{i9}$): the new version
5) ($t^-_{(i-1)7}$, $t^-_{i6}$), ($t^-_{(i-1)8}$, $t^-_{i6}$): login and signup with Microsoft accounts.
6) ($t^-_{(i-1)8}$, $t^-_{i9}$): contact list syncing and status update.

Interestingly, these points are verified by the short notes of Skype developers regarding their updates [21]. Specifically, the above topics can be associated with the following original developers claims: (a) General performance and reliability improvements (Version 2017.08.15, Version 2017.08.29: phone calls, video calls and messaging quality), (b) Improved sign in - sign back into your account more easily (Version 2017.08.15: "log in" features, user account related functions), (c) New controls added to help users manage vibration and LED notification alerts. (Version 2017.07.05: notification), (d) Improvements to PSTN call stability (Version 2017.07.05: connection), (e) Messaging improvements  Add content to chats via the + button and enjoy more room for your messages. (Version 2017.08.02: messaging), (f) The ability to add or remove contacts from your profile (Version 2017.08.01), (g) Activity indicators - see who's currently active in your Chats list (Version 2017.08.02: contacts and statuses). Hence, due to the correlation between the previously mentioned issues detected using our method and the content of the following up updates, we can verify the existence of those issues. However, whether the reason of the according update is the user reviews is unknown. On the other hand, we can also detect the disagreement amongst users' opinions. For example, a number of users think the calling quality deteriorated after the update while many others think it was improved (($t^+_{(i-1)6}$, $t^-_{i10}$) and ($t^-_{(i-1)3}$, $t^+_{i6}$)). A number of users also think the update improves the app when other users think it is just as bad as the previous (($t^-_{(i-1)7}$, $t^+_{i3}$) and ($t^-_{(i-1)3}$, $t^-_{i10}$)). We can obtain more details regarding users' different opinions by further investigating the keywords-related review texts.

 ISBN: 978-1-61208-668-2

## IV. Related Work

The spontaneous feedback from the users, i.e., the user reviews, helps effectively the evolution of the target system, where a key to enable such feedback is to ease the users effort in composing and uploading it [22]. For the contemporary mobile apps, the app distribution platforms, e.g., Apple App-Store, and Google Play, enable such spontaneous reviews of the users and facilitate the developers in terms of maintaining and evolving mobile apps [23]. Such feedback and reviews contain helpful information for developers in terms of identifying missing features and improving software quality [24]. From those review information, user requirements can be elicited continuously and, in a crowd-based fashion [25] [26].

The reviews of the mobile app users reflect a variety of topics, which majorly cover the perspectives of bug reports, feature requests, user experiences, solution proposal, information seeking and giving, and general ratings [2]–[4].

For such purpose, NLP, SA, and supervised learning are the common techniques used to classify user reviews [3] [4]. According to recent studies, the combination of NLP and SA techniques has high accuracy in detecting useful sentences [4]. These techniques are also used in many other studies in terms of review opinion mining [7] [9] [24].

Many studies have contributed to the user opinion mining of mobile apps. Fu et al. [5] proposes WisCom, which can detect why the users like or dislike a particular app based on the users reviews throughout time and provide insights regarding the users concerns and preferences in the app market. Similar studies regarding mining information from app stores data focus on different perspectives of the issues, e.g., the correlations between ratings and download rank [6], ratings and API use [27], review classification and useful sentences detection [4]. On the other hand, Chen et al. [7] provides the AR-miner framework, which facilitates the informative reviews extraction, review grouping based on topics, review prioritization and informative reviews presentation with visualization. Guzman and Maalej [8] proposes an approach focusing on feature extraction and sentiment analysis, which facilitates the evaluation of individual app features. Similarly, Iacob and Harrison [9] focuses also on the feature requests extraction but via means of linguistic rules and LDA topic modeling. Many studies also provide methods of using automatic classification method to study mobile app user reviews [28] [29] [30]

Compared to the previous mentioned approaches in user review opinion mining, our method aims towards the similar topic detection and analysis concerning not only the app in general but also the particular major updates. Furthermore, we focus on the topic similarity of data segments, classified by sentiment analysis and supervised learning, which is different from the methods mentioned above. It enables the developers to acquire information regarding each individual update and will provide insights on the future updates.

## V. Conclusion

In this study, we propose a method for analyzing the correlation of mobile apps user reviews before and after a particular app updates in order to detect how users' opinions change with the update released. After classifying the reviews before and after a particular update by positive and negative sentiment, we extract the topics of each segment. By comparing the similarities of these extracted topics, we identify both the positive and negative issues reflected by these reviews regarding the particular update and the app in general. Overall, this study is an exploratory investigation on using user review opinion mining techniques in detecting update-specific issues. The future studies shall extend the use of this method to the whole maintenance lifecycle of mobile apps to investigate the broader correlation between users feedback and the apps' update trends. Releasing strategies improvement based on this method will also be studied. Other factors, e.g., the different app categories, different platforms, and different mining and analysis techniques will also be taken into account.

## References


[1] X. Li, Z. Zhang, and J. Nummenmaa, "Models for mobile application maintenance based on update history," in Evaluation of Novel Approaches to Software Engineering (ENASE), 2014 International Conference on. IEEE, 2014, pp. 1–6.

[2] H. Khalid, E. Shihab, M. Nagappan, and A. E. Hassan, "What do mobile app users complain about?" IEEE Software, vol. 32, no. 3, 2015, pp. 70–77.

[3] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? on automatically classifying app reviews," in 2015 IEEE 23rd international requirements engineering conference (RE). IEEE, 2015, pp. 116–125.

[4] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "How can i improve my app? classifying user reviews for software maintenance and evolution," in Software maintenance and evolution (ICSME), 2015 IEEE international conference on. IEEE, 2015, pp. 281–290.

[5] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. 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. ACM, 2013, pp. 1276–1284.

[6] M. Harman, Y. Jia, and Y. Zhang, "App store mining and analysis: Msr for app stores," in Proceedings of the 9th IEEE Working Conference on Mining Software Repositories. IEEE Press, 2012, pp. 108–111.

[7] N. Chen, J. Lin, S. C. Hoi, X. Xiao, and B. Zhang, "Ar-miner: mining informative reviews for developers from mobile app marketplace," in Proceedings of the 36th International Conference on Software Engineering. ACM, 2014, pp. 767–778.

[8] E. Guzman and W. Maalej, "How do users like this feature? a fine grained sentiment analysis of app reviews," in Requirements Engineering Conference (RE), 2014 IEEE 22nd International. IEEE, 2014, pp. 153–162.

[9] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in Proceedings of the 10th Working Conference on Mining Software Repositories. IEEE Press, 2013, pp. 41–44.

[10] C. H. E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in ICWSM, 2014.

[11] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," Journal of Machine Learning Research, vol. 3, 2003, pp. 993–1022.

[12] T. T. Tanimoto, "Ibm internal report," Nov, vol. 17, 1957, p. 1957.

[13] B. V. Kumar and L. Hassebrook, "Performance measures for correlation filters," Applied optics, vol. 29, no. 20, 1990, pp. 2997–3006.

[14] Langdetect, https://pypi.python.org/pypi/langdetect, last accessed on 06/20/18.

[15] Pyenchant, https://pypi.python.org/pypi/pyenchant, last accessed on 06/20/18.

[16] NLTK, http://www.nltk.org, last accessed on 06/20/18.

[17] E. Cambria, R. Speer, C. Havasi, and A. Hussain, "Senticnet: A publicly available semantic resource for opinion mining." in AAAI fall symposium: commonsense knowledge, vol. 10, no. 0, 2010.

[18] S. Baccianella, A. Esuli, and F. Sebastiani, "Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining." in Lrec, vol. 10, no. 2010, 2010, pp. 2200–2204.


 ISBN: 978-1-61208-668-2 

[19] M. M. Bradley and P. J. Lang, "Affective norms for english words (anew): Instruction manual and affective ratings," Citeseer, Tech. Rep., 1999.

[20] M. Stevenson and Y. Wilks, "Word sense disambiguation," The Oxford Handbook of Comp. Linguistics, 2003, pp. 249–265.

[21] "Skype on Google Play," URL: https://play.google.com/store/apps/details?id=com.skype.raider, last accessed on 06/20/18.

[22] K. Schneider, "Focusing spontaneous feedback to support system evolution," in RE, pp. 165–174.

[23] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," in Requirements Engineering Conference (RE), 2013 21st IEEE International. IEEE, 2013, pp. 125–134.

[24] L. V. Galvis Carreño and K. Winbladh, "Analysis of user comments: an approach for software requirements evolution," in Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, 2013, pp. 582–591.

[25] N. Seyff, F. Graf, and N. Maiden, "Using mobile re tools to give end-users their own voice," in Requirements Engineering Conference (RE), 2010 18th IEEE International. IEEE, 2010, pp. 37–46.

[26] E. C. Groen, J. Doerr, and S. Adam, "Towards crowd-based requirements engineering a research preview," in International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, 2015, pp. 247–253.

[27] G. Bavota, M. Linares-Vasquez, C. E. Bernal-Cardenas, M. Di Penta, R. Oliveto, and D. Poshyvanyk, "The impact of api change-and fault-proneness on the user ratings of android apps," IEEE Trans. on Soft. Engin., vol. 41, no. 4, 2015, pp. 384–407.

[28] S. McIlroy, N. Ali, H. Khalid, and A. E. Hassan, "Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews," Empirical Software Engineering, vol. 21, no. 3, 2016, pp. 1067–1106.

[29] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," Requirements Engineering, vol. 21, no. 3, 2016, pp. 311–331.

[30] Z. Sun, Z. Ji, P. Zhang, C. Chen, X. Qian, X. Du, and Q. Wan, "Automatic labeling of mobile apps by the type of psychological needs they satisfy," Telematics and Informatics, vol. 34, no. 5, 2017, pp. 767–778.

 ISBN: 978-1-61208-668-2 

# PUBLICATION III

**Mobile App Evolution Analysis Based on User Reviews**

X. Li, Z. Zhang and K. Stefanidis

*International Conference on Intelligent Software Methodologies, Tools, and Techniques (SoMeT)*, 2018, 773–786

*September 2018*

# Mobile App Evolution Analysis based on User Reviews

Xiaozhou LI [a,1], Zheying ZHANG [a] and Kostas STEFANIDIS [a]
[a] *University of Tampere, Finland*

**Abstract.** The user reviews of mobile apps are important assets that reflect the users' needs and complaints about particular apps regarding features, usability, and designs. From investigating the content of such reviews, the app developers can acquire useful information guiding the future maintenance and evolution work. Previous studies on opinion mining in mobile app reviews have provided various approaches to eliciting such critical information. A particular update of an app can provide changes to the app that result in users' reversed opinions, as well as, specific new complaints or praises. However, limited studies focus on eliciting the user opinions regarding a particular mobile app update, or the impact the update imposes. In this paper, we propose a method for systematically studying and analyzing the evolution of the users' opinions taking into consideration a set of mobile app updates. For doing so, we compare the topics appearing in the users' reviews before and after the updates. We also validate the method with an experiment on an existing mobile app.

## 1. Introduction

The increasing number of smartphone users has led to a continuous increase in the number of mobile apps and their overall usage. Users browse and download apps via different digital distribution platforms, and these platforms also provide an important channel for users to provide feedback to the app developers. Users can rate a release and express their opinions on the release they are using at the time of submitting a review, as in traditional recommender systems [13, 16]. Along with frequent release updates through the distribution platforms, the reviews accumulate the bugs, desired features, the comparison and users' attitude toward the app informally, and form a useful resource for analyzing change requests in the app maintenance phase.

Existing studies have proposed different approaches to identifying changes [22, 26]. One particular example is to identify change requests from user reviews for mobile app maintenance. With various opinion mining techniques, such as natural language processing (NLP), sentiment analysis (SA) and supervised learning, many studies have been conducted regarding the classification of reviews towards different issue perspectives [19, 21]. Other perspectives, such as user preferences, entities analysis, app evaluation, user satisfaction, the relation between download and rating, feature extraction and review prioritization, have also been widely studied [5, 7, 8, 12, 14, 15, 18]. However, lim-

[1] Corresponding Author: University of Tampere, Finland; E-mail: xiaozhou.li@uta.fi

ited studies focus on the use of such methods in opinion mining on particular updates of a mobile app and the impact on the app's updates in the following releases, despite the importance of such information. It is unclear how bugs and requests in user reviews are addressed in the follow-up releases and how users' attitude towards a particular issue changes when new updates are released.

In this paper, we investigate the correlation between users' positive and negative reviews before and after a sequence of apps' releases. We consider two dimensions: time and sentiment. Specifically, we divide user reviews based on a set of major updates and distinguish them between those precede and follow the particular updates. Each group of reviews is further divided into positive and negative ones using sentiment analysis. We propose a way to measure the similarity of each group of reviews. The measurement reveals the similarity and changes between different groups of reviews and helps to gain insight into the app's change requests in the maintenance phase. Besides understanding the main issues in user reviews, it helps developers be aware of the users' opinions on particular updates and guides them proactively to address the most important issues early.

## 2. Method Description

### *2.1. Preliminaries*

This method aims to detect the evolving topic trend regarding a particular app by investigating the correlation between its users' positive and negative reviews before and after each major update within a sequence of updates through the app maintenance life cycle. Such correlation shall show the degree to which the users' comments are reflected in the sequence of updates and such updates are accepted by the users. The factors that influence and reflect such correlation include the main topics of the reviews between every two updates, the sentiment of those reviews, and the topics similarities before and after each update. Next, we illustrate how to detect the correlation via investigating these factors.

Let $R$ be a collection of user reviews for a particular mobile app $A$, covering the time period $T_{time} = [ti_s, ti_e]$ (where $ti_s, ti_e$ are two different time points with $ti_s < ti_e$). Therein, each review $r_i \in R$ is associated with a particular time when $r_i$ is published. Let $U$ be the set of updates released by the app developers within the time period $T_{time}$. Meanwhile, each update is also associated with a time point $t_i$, reflecting the time $u_i$ is announced and delivered. Furthermore, we consider the review set $R_i$, of which each $r_i$ is published within $[t_i, t_{i+1}]$, as the collection of reviews regarding the update $u_i$. Hence, for the $n$ updates $\{u_i | i \in N, k \leq i < k+n\}$ where $k \geq 1$ and $n > 0$ for the app $A$ within period T, the review set $R$ can be divided into $n+1$ subset where each $R_i \subseteq R$ is the set of reviews commenting on the according $u_i$. $R_0$ is the review set given before the first update of A. For each review sentence $r_j \in R_i$, a sentiment score shall be calculated and assigned to $r_j$, whose sentiment is either *positive* or *negative*. In this way, by identifying the sentiment of each individual review in $R_i$, we can divide $R_i$ into a positive review set $R_i^+$ and a negative one $R_i^+$, where $R_i^+ \cup R_i^- = R_i$ and $R_i^+ \cap R_i^- = \emptyset$.

When modeling topics of each review set, we assign $T_i^+$ and $T_i^-$ as the topic set for the positive and negative reviews. Therefore, we investigate the merits and issues of a particular update $u_i$ by compare the similarities and changes between $T_{i-1}^+$, $T_{i-1}^-$, $T_i^+$, and

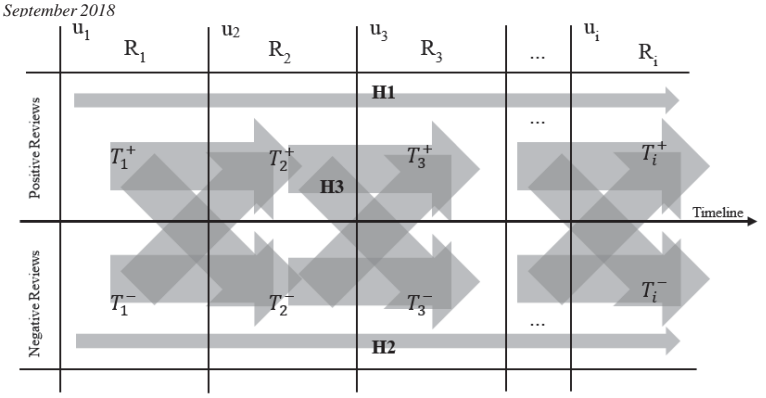


**Figure 1.** Hypotheses for the updates and the topic change of user reviews.

$T_i^-$. Furthermore, by investigating such topic similarity and changes regarding each $u_i$, we shall be able to observe the users' opinion change on the merits and issues within a given period of time. Therefore, given a set of updates U = $\{u_1, u_2, ... u_n\}$ with the according review sets $R_1^+, R_1^-, R_2^+, R_2^-, ... R_n^+, R_n^-$, finding the topic similarity correlations throughout $T_1^+, T_1^-, T_2^+, T_2^-, ... T_n^+, T_n^-$ shall reflect the evolving user opinions concerning various aspects of the app within a particular period of time. Therefore, provided a set of topic $\{t_1, t_2, ... t_i ... t_n | t_i \in T_i\}$, where each $t_i$ is similar to $t_{i+1}$ ($1 \leq i < n$), then such topic set is defined as a *similar topic chain*. We furthermore propose three hypotheses as follows.

- H1. The similar topic chains through $T_i^+, T_{i+1}^+, ... T_n^+$ reflect the merits and users' praise regarding a sequence of app $A$'s updates $u_i, u_{i+1}, ... u_n$.
- H2. The similar topic chains through $T_i^-, T_{i+1}^-, ... T_n^-$ reflect the issues and users' complaints regarding a sequence of app $A$'s updates $u_i, u_{i+1}, ... u_n$.
- H3. The similar topic chains containing topics from both $T_i^+, T_{i+1}^+, ... T_n^+$ and $T_i^-, T_{i+1}^-, ... T_n^-$ reflect the changing user opinions (positive to negative, or negative to positive) regarding particular aspects of app $A$.

Thus, the results obtained from these hypotheses for a certain period of updates $u_i, u_{i+1}, ... u_n$ provide the following information: 1) the merits and users' praise for the app $A$, 2) the issues and users' complaints for the app $A$, and 3) the evolving of users opinions regarding particular aspects of the app $A$.

### *2.2. Sentiment Classification*

The aim of sentiment classification in this method is to classify each review set $R_i$ into two subsets, i.e., $R_i^+$ and $R_i^-$. Herein, $R_i^+$ denotes the set of positive reviews from $R_i$, and $R_i^-$ denotes the set of negative reviews. Therefore, each $r_j$ in $R_i$ shall be determined whether it is positive or negative.

To do so, we assign a sentiment score to each review by exploiting a robust tool for sentiment strength detection on social web data. As each $r_j$ can be seen as a list of words $W_j$, we first select a lexicon that will determine the sentiment score of each word $w_z$ in $W_j$. The lexicon for sentiment analysis is a list of words used in the English language, each of which is assigned with a sentiment value in terms of its sentiment valence (intensity) and

polarity (positive/negative). To determine the sentiment of words, we assign a rational value within a range to a word. For example, if the word "okay" has a positive valence value of 0.9, the word "good" must have a higher positive value, e.g., 1.9, and the word "great" has even higher value, e.g., 3.1.

Furthermore, the lexicon set shall include social media terms, such as Western-style emoticons (e.g., :-)), sentiment-related acronyms and initialisms (e.g., LOL, WTF), and commonly used slang with sentiment value (e.g., nah, meh).

With the well-established lexicon and a selected set of proper grammatical and syntactical heuristics, we shall then be able to determine the overall sentiment score of a review. Namely, the sentiment score of a review $r_j$ is equal to $S_j$, where $S_j \in (-1,1)$. The grammatical and syntactical heuristics are seen as the cues to change the sentiment of word sets. Therein, punctuation, capitalization, degree modifier, and contrastive conjunctions are all taken into account. For example, the sentiment of "The book is EXTREMELY AWESOME!!!" is stronger than "The book is extremely awesome", which is stronger than "The book is very good.".

With both the lexicon value for each word of the review and the calculation based on the grammatical and syntactical heuristics, we can then assign unique sentiment values to each review. That is, each review $r_j$ is classified into positive, neutral or negative, as follows:

$$r_j \text{ is } \begin{cases} \text{positive, if } 0 < S_j < 1, \\ \text{neutral, if } S_j = 0, \\ \text{negative, if } -1 < S_j < 0. \end{cases}$$

Overall, each review set $R_i$ is divided into $R_i^+$, $R_i^0$, and $R_i^-$, denoting the positive, neutral and negative review sets. By experimentally observing, $R_i^0$ contains a large number of reviews with also useful information. Therefore, we further classify $R_i^0$ into positive and negative using the Naive Bayes Classifier with the training data from $R_i^+$ and $R_i^-$. $R_i^0$ is classified into $R_i^{0+}$ and $R_i^{0-}$, which in turn, are added to $R_i^+$ and $R_i^-$, respectively. The reason to perform supervised classification after sentiment analysis instead of directly applying classification is twofold. Firstly, manually creating training data is time-consuming and less accurate than using existing sentiment analysis methods. Secondly, training the sentiment classified reviews will provide domain specific and reliable results.

### *2.3. Topic Analysis*

After dividing the review sets $R_i$ into $R_i^+$ and $R_i^-$, we elicit the main topics from each of the classified review sets by exploiting the Latent Dirichlet Allocation (LDA) method [3]. First, we consider each review sentence $r_j$ in a particular set of reviews as a list of words $W_j$, where the sequence of the words is not recorded. The number of topics in this review set is set as $t$. Presumably, there is a distribution for the probability of a particular word appears in a particular topic, when there is also one for that of a particular review of a topic. We build the set of $Review-Topic$, where each word of each review is assigned with a topic out of the $t$ topics. As preparation, we define $Review-topic-numbers$, $Topic-words-numbers$, and $Topic-numbers$ denoting the number of occurrence of each topic in each review, the number of occurrence of each

word in each topic, and the number of words in each topic, respectively. For example, $Review-topic-numbers(r_j,k)$ denotes the number of occurrences of topic $k$ in review $r_j$. Then, we randomly assign each word $w_z$ of each review $r_j$ with a topic $t_k$. Accordingly, the $Review-topic-numbers$, $Topic-words-numbers$, and $Topic-numbers$ will be updated as the referencing weight of the distribution of the words for each topic. Then iteratively, for each word, we assign a new topic based on such weight of distribution and adjust the weight with the $Review-topic-numbers$, $Topic-words-numbers$, and $Topic-numbers$ for the next iteration. After a given number of iteration, $t$ topics will be determined by the $Topic-words-numbers$, which is the number of occurrences of the words in each topic. Each $t_k$ is then denoted by the most common keywords used in this topic. Then, the set of topics $T$ is returned as a result. For the review sets $R_i^+$, $R_i^-$, $R_{i-1}^+$ and $R_{i-1}^-$, we will have the topics sets $T_i^+$, $T_i^-$, $T_{i-1}^+$ and $T_{i-1}^-$ accordingly.

### *2.4. Calculating Topics Similarities*

Based on the topic sets $T_i^+$ and $T_i^-$ elicited from the review sets $R_i^+$ and $R_i^-$,we further analyze the similarities between the individual topics between each pair of the topic sets. As the result from the previous topic analysis, each topic set $T$ encompasses $k$ topics, each of which is represented by the list of the most possible appearing keywords. Thus, each topic set $T$ with $k$ topics each of which is represented by $w$ keywords, can be denoted as:

$$T = \begin{bmatrix} kw_{1,1} & kw_{1,2} & \dots & kw_{1,w} \\ \dots & \dots & \dots & \dots \\ kw_{k,1} & kw_{k,2} & \dots & kw_{k,w} \end{bmatrix}$$

with each $t_i \in T$ can be denoted as $[kw_{i,1}, kw_{i,2}, ...kw_{i,k}]$. To compare the similarity between two topic sets, each consisting of $k$ topics, we compare all pairs of topics. Due to the fact that each topic is represented as a set of keywords, the similarity of two topics shall be denoted by the common keywords of these topics. Hence, an easy way for calculating the similarity between any two topics $t_i$ and $t_j$ is by using the Jaccard similarity. This similarity function reflects the percentage of the common keywords of the two sets in the whole keywords set of the two: $J(t_i,t_j) = \frac{|t_i \cap t_j|}{|t_i \cup t_j|}$.

On the other hand, the probability ($p_{ij}$) of each keyword $kw_{ij}$ appearing in topic $t_i$ is different. Hence, despite J($t_i$, $t_j$) = J($t_i$, $t_k$), when the probability value of the common keywords varies, the topic similarities for $t_j$, $t_k$ towards $t_i$ can still differ. Therefore, we propose a modification of the previously described Jaccard similarity calculation taking into account the probability of the keywords, named as the Jaccard Extended Similarity. The $c$ common keywords of $t_i$ and $t_j$ include $[kw_{ij,1}, kw_{ij,2}, ...kw_{ij,c}]$, when the according probability list in $t_i$ and $t_j$ is $[p_{i,1}, p_{i,2}, ...p_{i,c}]$ and $[p_{j,1}, p_{j,2}, ...p_{j,c}]$. The Jaccard Extended Similarity is then calculated as follows:

$$JE(t_i,t_j) = \frac{\sum_{x=1}^{c} \frac{p_{i,x}+p_{j,x}}{2}}{\sum_{x=1}^{c} \frac{p_{i,x}+p_{j,x}}{2} + \sum_{x=c+1}^{k} p_{i,x} + \sum_{x=c+1}^{k} p_{j,x}}.$$

The probability for each keyword of any topic belongs to (0,1). Hence, for this formula, when $t_i$ and $t_j$ contain more common keywords, the numerator increases mono-

tonically, and the denominator decreases monotonically. Therefore, $JE(t_i,t_j)$ increases when $t_i$ and $t_j$ have more keywords in common. In addition, when the probability of the common keywords increases, $\sum_{x=1}^{c} \frac{p_{i,x}+p_{j,x}}{2}$ increases. Because the denominator is greater than the numerator, and both are greater than 0, $JE(t_i,t_j)$ increases when the probabilities of the common keywords of $t_i$ and $t_j$ increase. In this way, JE takes into account both the number of common keywords and the probabilities of such keywords.

### *2.5. Identifying Matching Topics*

After computing similarities between pairs of review topics, we shall also identify which are the matching topics when cross-comparing the topics of the topics sets $T_i^+$, $T_i^-$, $T_{i-1}^+$ and $T_{i-1}^-$. Hence, to identify the matching topics between two review topic sets $T_a$ and $T_b$, the aim is to identify all the topic pairs $(t_{ai}, t_{bj})$, $t_{ai} \in T_a$ and $t_{bj} \in T_b$, that have the highest similarity, and contain the unique $t_{ai}$ and $t_{bj}$ which are both different from those of the other selected pairs. We set a threshold similarity value, so that only the topic pairs that have a similarity value greater than it can be considered as similar. Given that several topics are similar to multiple other topics, we apply an algorithm to select the unique similar topic pairs when comparing two review topic sets, $T_a$ and $T_b$.

**Algorithm 1** Matching Topics Identification

**procedure** MATCHING TOPICS IDENTIFICATION
$k \leftarrow$ number of topics
$Ta \leftarrow \{t_{a1}, t_{a2}, ..., t_{ai}, ..., t_{ak}\}$
$Tb \leftarrow \{t_{b1}, t_{b2}, ..., t_{bj}, ..., t_{bk}\}$
$s_{ij} \leftarrow Similarity(t_{ai}, t_{bj})$
*For* $i, j \in \{1, 2, ..., k\}$
    **if** $s_{ij} \geq$ threshold value **then**
        $S \leftarrow s_{ij}$
*While* $len(S) > 0$
    **if** Only 1 Max(S) in S **then**
        P.append($(t_{ax}, t_{by})$) where $Similarity(t_{ax}, t_{by}) = Max(S)$
        S - (S $\cap$ ($\{s_{x1}, s_{x2}, ..., s_{xk}\} \cup \{s_{1y}, s_{2y}, ..., s_{ky}\}$))
    **if** More than 1 Max(S) in S **then**
        P.append $((t_{ax}, t_{by}))$ where $Similarity(t_{ax}, t_{by}) = Max(S)$ & $Sum(\{s_{x1}, ..., s_{xk}, s_{1y}, ..., s_{ky}\})$is the minimum
Return P

The aim of Algorithm 1 is to select only the similar topic pairs with the highest similarity value. When a particular topic pair is selected, the other pairs which either of these two selected topics is also pairing with will be ignored. With this pair and the ignored ones deducted from the original selected similar topic pairs, we continuously select the one with highest similarity value, until all selected pairs are unique without any selected topic being simultaneously similar to more than one topic. Especially, if at a certain selecting iteration, there is a tie on the highest similarity value, we select the pair of topics that are the least similar to the rest of the topics, by calculating the sum of all the similarity values with either topic and selecting the one with the minimum sum value.

## 3. Case Study

### *3.1. Preprocessing*

Before starting the experiment with the proposed method, preprocessing on the raw review data is required. The whole preprocessing work can be divided into four individual steps as follows.

**Filtering non-English reviews.** The raw review data may contain a number of review items that are not written in English, which needs to be filtered out. Also, similar to social media text, user reviews usually contain many commonly used slurs that are not regular English vocabularies. Our goal is to not filter out these words, as they likely contain sentiment related information, without which shall influence our experiment results. Overall, we screen out the non-English review sentences using Langdetect[2], a convenient language detecting package for Python language. Compared with PyEnchant[3], another language detecting package, Langdetect enables determining the language of text on sentence level. It shall remain the review data containing such English slurs.

**Focusing on sentence-level granularity.** Because each user review can contain more than one sentence, a multi-sentence review can contain multiple meanings, one for each sentence. Thus, we divide each review from a review set $R_i$ into individual sentences. Hereafter, we use $r_j$ to denote a review sentence in $R_i$. We use the sentence tokenizer feature from the NLTK[4] python package, with a further checking on the legitimacy of the sentences.

**Sentiment Classification with VADER.** To perform sentiment analysis on the collected app reviews, we select the Valence Aware Dictionary for sEntiment Reasoning (VADER) approach [10]. Compared with other sentiment analysis tools, VADER has a number of advantages regarding this study. Firstly, the classification accuracy of VADER on sentiment towards positive, negative and neutral classes is even higher than individual human raters in social media domain. In addition, its overall classification accuracies on product reviews from Amazon, movie reviews, and editorials from NYTimes also outperform other sentiment analysis approaches, such as SenticNet [6], SentiWordNet [1], Affective Norms for English Words [4], and Word-Sense Disambiguation [25], and run closely with the accuracy of an individual human. On the other hand, VADER approach is integrated into the NLTK package, which can be easily imported and performed using Python.

**Filtering stop-words and non-verb-or-nouns and Stemming.** In addition, the collected English review sentences are also transformed into lower cases, screened with stop-words, and stemmed before topic modeling. In order to obtain more meaningful topic modeling results, we add the words that connect to only general information but have a significant appearing rate in the reviews to the list of stop words. For example, the name of the app and the word app are of neither help towards topic modeling nor towards sentiment analysis. Furthermore, because the sentiment analysis is done previously, we select only the nouns and verbs in the review sentence sets to enhance the analysis result.

[2] https://pypi.python.org/pypi/langdetect
[3] https://pypi.python.org/pypi/pyenchant/
[4] http://www.nltk.org

### *3.2. Datasets*

In this case study, we use the reviews submitted between 2016-09-13 and 2017-08-31 on the mobile app Whatsapp[5] on Android platform. We collected 1,148,032 reviews, which are then tokenized into 1,327,504 individual sentences. After eliminating the non-English review sentences, we have 1,012,088 English review sentences as experiment data.

We investigate the merits and issues concerning the major updates of Whatsapp released on Android platform. Within this period, 46 updates were released. On average, the app has been updated nearly every 7 - 8 days with one major update in every 50 days. By observing the content of each update from the given information on Google Play, we find that some consecutive updates contain the same content based on their descriptions. Therefore, we consider the first update of a set of updates which contain same descriptions as a major update, with the rest of the updates as minor updates. Accordingly, seven major updates are identified during the given period. They divided the review dataset into seven parts. Each sub review set is seen as the reviews regarding one particular update. Amongst the major updates of Whatsapp during this period, each update $\{u_1, u_2 \ldots u_7\}$ provides unique changes in UI design and user experiences. By classifying all selected review sentences into positive and negative using sentiment analysis and supervised classification with Naive Bayes Classifier, the number of review sentences in each segment is listed in Table 1. Accordingly, the number of reviews for each review sets of $R_1^+ \sim R_7^+$ and $R_1^- \sim R_7^-$, are shown in the figure below. Overall, the total number of positive reviews around the particular update is bigger than the number of negative reviews. Meanwhile, the monthly review number increased sharply after this particular major update.

**Table 1.** The number of Positive and Negative reviews after each Update

| | $u_1$ | $u_2$ | $u_3$ | $u_4$ | $u_5$ | $u_6$ | $u_7$ | |
|---|---|---|---|---|---|---|---|---|
| Release Date | 16.9.13 | 16.10.11 | 16.12.1 | 17.3.14 | 17.4.18 | 17.5.17 | 17.7.13 | |
| Positive | 39241 | 92222 | 246384 | 99889 | 50205 | 89766 | 72960 | 690667 |
| Negative | 15238 | 31674 | 165723 | 38174 | 18254 | 27646 | 24712 | 321421 |
| Sum | 54479 | 123896 | 412107 | 138063 | 68459 | 117412 | 97672 | 1012088 |

After sentiment analysis and classification, we perform the LDA topic analysis to identify the topics of the review set $R_1^+ \sim R_7^+$ and $R_1^- \sim R_7^-$, in order to investigate the users' opinions concerning the update and further verify the previously proposed hypothesis. To train the LDA topic models, we need to set the number of topics $k$. Based on an experimentally study regarding the quality of the topics produced for different $k$ values, and select $k = 10$. Overall, for the collected 1,012,088 review sentences on Whatsapp, we perform an LDA topic analysis on the review set. For each of the 14 sets of review data, we use the Gensim[6] topic modeling toolbox to train the 10-topic LDA models. For the 14 LDA models, each individual topic is represented by 20 keywords. Then, we assign each review sentence in the LDA models to the topic to which it has the highest probability to belong.

[5]https://www.whatsapp.com/

[6]https://radimrehurek.com/gensim/

### *3.3. Topic Evolution Analysis with Similar Topic Chains*

Each topic contains 20 keywords. For every major update $u_i$, there are four sets of associated reviews which results in four sets of topics, i.e., $T^+_{i-1}$, $T^-_{i-1}$, $T^+_i$, and $T^-_i$. We assess the similarity of topics after the update $u_i$, which provides 400 topics pairs. The number of common keywords for those 400 pairs range from 0 - 14. Since it is difficult to identify the topic content based on less than five keywords, we set an initial threshold equal to 0.143 for the Jaccard Extended similarity value, meaning that we consider two topics as similar only when they have five or more common keywords with the probability of these keywords above average. Based on this threshold, we apply Algorithm 1 to select the unique similar topic pairs in each of the four comparison sets (before and after each update, positive and negative reviews). Part of the results are shown in Figure 2. In Figure 2, positive topics are illustrated as blue circle while negative topic as red square. For example, $t^+_{1,7}$ is denoted as a blue circle marked "7" in column "T1" when $t^-_{2,10}$ is denoted as a red square marked "10" in column "T2".

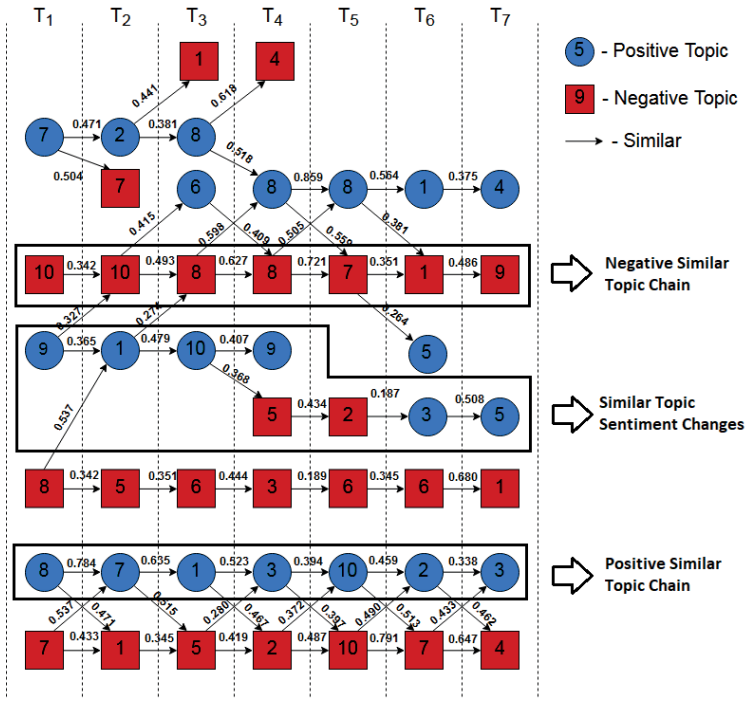


**Figure 2.** A Segment of the Topic Similarity Graph

We can firstly analyze the topic similarity between two sets of reviews. Based on the common keywords of each identified similar topic pairs, we could extract the users'

opinions regarding the particular update. For example, in Figure 2, we can identify that $t^+_{1,7}$, $t^+_{1,9}$, and $t^+_{1,8} \in T_1^+$ are similar to $t^+_{2,2}$, $t^+_{2,1}$, and $t^+_{2,7} \in T_2^+$ respectively. The common keywords for these three topic pairs are listed in Table 2. Therefore, these similar topic pairs suggest that the users reflect positively regarding the four topics both before and after Update 2 (Version 16.10.11), including the quality of video and voice call, asking for fixing and updating themes and emoji feature, and, contacts and social group option with picture sending feature. Similarly, by identifying the similar topic pairs ($t^-_{1,10}$, $t^-_{2,10}$), ($t^-_{1,8}$, $t^-_{2,5}$), and ($t^-_{1,7}$, $t^-_{2,1}$), we can also conclude that the users are still not happy regarding the function of contact group and its options, message sending and receiving, and the quality of video calls after Update 2. Similarity, by analyzing the common keywords between positive topics and negative topics, such as, ($t^+_{1,9}$, $t^-_{2,10}$) and ($t^-_{1,7}$, $t^+_{2,7}$) in Figure 2, we can also identify the users' opinion changes after Update 2. The common keywords of topic pair ($t^+_{1,9}$, $t^-_{2,10}$) reflect that the users start to complain negatively concerning the feature of sending image and picture in chat group after the update. The common keywords of ($t^-_{1,7}$, $t^+_{2,7}$) then reflects users become more satisfied with the general call feature as well as the video and voice quality.

**Table 2.** Common keywords of Similar Topic Pairs Regarding Update 2

| Topic Pairs | Common Keywords |
|---|---|
| ($t^+_{1,7}$, $t^+_{2,2}$) | ['chang', 'emoji', 'fix', 'plea', 'pls', 'theme', 'updat', 'want'] |
| ($t^+_{1,9}$, $t^+_{2,1}$) | ['contact', 'group', 'option', 'photo', 'pic', 'pictur', 'send', 'share', 'want'] |
| ($t^+_{1,8}$, $t^+_{2,7}$) | ['ad', 'add', 'call', 'facil', 'featur', 'make', 'need', 'option', 'qualiti', 'video', 'voic'] |
| ($t^-_{1,10}$, $t^-_{2,10}$) | ['add', 'contact', 'group', 'list', 'option', 'want'] |
| ($t^-_{1,8}$, $t^-_{2,5}$) | ['come', 'get', 'messag', 'open', 'phone', 'read', 'receiv', 'see', 'seen'] |
| ($t^-_{1,7}$, $t^-_{2,1}$) | ['call', 'featur', 'option', 'qualiti', 'video', 'work'] |
| ($t^+_{1,9}$, $t^-_{2,10}$) | ['contact', 'group', 'imag', 'list', 'option', 'pictur', 'send', 'status', 'want'] |
| ($t^-_{1,7}$, $t^+_{2,7}$) | ['call', 'featur', 'make', 'need', 'option', 'qualiti', 'video', 'voic', 'work'] |

Furthermore, by investigating the similar topic pairs throughout the evolution timeline of a particular app, we can detect the evolution trend of the app. For example, from Update 1 to Update 7, the positive opinion regarding each update can be reflected by the common keywords of $T_1^+$ and $T_2^+$ to $T_6^+$ and $T_7^+$. Therefore, the merit of the app throughout these updates can be reflected by the similar topic chain shown in Table 3.

**Table 3.** Common keywords in a Positive Similar Topic Chain

| Topic Pairs | Common Keywords |
|---|---|
| ($t^+_{1,8}$, $t^+_{2,7}$) | ['ad', 'add', 'call', 'facil', 'featur', 'make', 'need', 'option', 'qualiti', 'video', 'voic'] |
| ($t^+_{2,7}$, $t^+_{3,1}$) | ['call', 'facil', 'make', 'need', 'qualiti', 'vedio', 'video', 'voic'] |
| ($t^+_{3,1}$, $t^+_{4,3}$) | ['call', 'improv', 'make', 'need', 'qualiti', 'video', 'voic'] |
| ($t^+_{4,3}$, $t^+_{5,10}$) | ['call', 'data', 'improv', 'make', 'qualiti', 'send', 'video', 'voic'] |
| ($t^+_{5,10}$, $t^+_{6,2}$) | ['call', 'group', 'improv', 'make', 'qualiti', 'send', 'video', 'voic'] |
| ($t^+_{6,2}$, $t^+_{7,3}$) | ['call', 'featur', 'group', 'make', 'send', 'video', 'voic'] |

From the similar topic pairs identified in Figure 2, we could claim that topic $t^+_{1,8}$, $t^+_{2,7}$, $t^+_{3,1}$, $t^+_{4,3}$, $t^+_{5,10}$, $t^+_{6,2}$, and $t^+_{7,3}$ are all identified as similar to one another. The similarity is also reflected by the common keywords listed in Table 3, based on which we could conclude

from Update 2 to Update 7, the users constantly share positive opinions regarding the video and voice call quality of the app, which verifies Similarly, the constant negative opinions can be reflected by the common keywords of the similar negative topic chains throughout updates. For example, topic $t^-_{1,10}$, $t^-_{2,10}$, $t^-_{3,8}$, $t^-_{4,8}$, $t^-_{5,7}$, $t^-_{6,1}$, and $t^-_{7,9}$ are identified as similar (shown in Table 4). Such negative topic similarity chain reflects the users' request for features regarding options of contact and status.

**Table 4.** Common keywords in a Negative Similar Topic Chain

| Topic Pairs | Common Keywords |
|---|---|
| ($t^-_{1,10}$, $t^-_{2,10}$) | ['add', 'contact', 'group', 'list', 'option', 'want'] |
| ($t^-_{2,10}$, $t^-_{3,8}$) | ['add', 'contact', 'list', 'option', 'pictur', 'remov', 'see', 'status', 'want'] |
| ($t^-_{3,8}$, $t^-_{4,8}$) | ['add', 'featur', 'need', 'option', 'photo', 'put', 'remov', 'status', 'stori', 'text'] |
| ($t^-_{4,8}$, $t^-_{5,7}$) | ['add', 'featur', 'option', 'plea', 'remov', 'status', 'stori', 'text'] |
| ($t^-_{5,7}$, $t^-_{6,1}$) | ['featur', 'option', 'peopl', 'status', 'want'] |
| ($t^-_{6,1}$, $t^-_{7,9}$) | ['contact', 'featur', 'list', 'peopl', 'see', 'seen', 'status', 'view'] |

By investigating the update history of Whatsapp, we we can observe that the "*video calling*" feature was added to Whatsapp at Update 3 (Version 16.12.1). This can explain why the common keyword '*call*' became the highest probable common keyword from ($t^+_{2,7}$, $t^+_{3,1}$). On the other hand, Whatsappp added one update on "*WhatsApp Status: Post photos, videos, and GIFs to your status and share with your contacts what's going on throughout your day. Status updates from your contacts appear in the Status tab, and they'll disappear after 24 hours. Long press on a contact's name in the Status tab to mute their updates.*" in Update 4 (Version 17.3.14) and another update "*You can now send multiple contact cards at once*" in Update 5 (Version 17.4.18). This partially proves the developers noticed the constant negative opinions of users shown from Table 4

Moreover, in Figure 2, we can also identify particular topics are considered similar to one positive topic and one negative topic from the next review set simultaneously. For example, $t^+_{3,10}$ is similar to both $t^+_{4,9}$ and $t^-_{4,5}$. From the common keywords of topic pair ($t^+_{2,1}$, $t^+_{3,10}$), we can observe that the users share positive opinion regarding the feature of contacting and chat groups. Then topic pair ($t^+_{3,10}$, $t^+_{4,9}$) contains common keywords of ['add', 'featur', 'group', 'make', 'option', 'person', 'provid', 'show'], when ($t^+_{3,10}$, $t^-_{4,5}$) contains common keywords of ['ad', 'contact', 'group', 'list', 'peopl', 'see', 'seen', 'show']. These keywords suggest that after Update 4, a number of users are satisfied with the chat group feature and options, when some users are not happy with the contact and chat list. After Update 5, topic pair ($t^-_{4,5}$, $t^-_{5,2}$) suggests that the complaints regarding contact and chat list continues until Update 6 where ($t^-_{5,2}$, $t^+_{6,3}$) suggests that the complaints regarding contact and chat list is not significant any more. By investigating the update history of Whatsapp, we can observe that in Update 6 (Version 17.5.17) the developers added a feature "*Pin chats to the top of your chat list, so you can quickly find them. Just tap and hold on a chat and tap the pin icon at the top of your screen*", which results in the reduce in negative opinions regarding the chat list feature. It also verifies the existence of the significant complaints before this update regarding this matter.

## 4. Related Work

The spontaneous feedback from the users, i.e., the user reviews, helps effectively the evolution of the target system, where a key to enable such feedback is to ease the users' effort in composing and uploading it [23]. For the contemporary mobile apps, the app distribution platforms, e.g., Apple AppStore, and Google Play, enable such spontaneous reviews of the users and facilitate the developers in terms of maintaining and evolving mobile apps [20]. Such feedback and reviews contain helpful information for developers in terms of identifying missing features and improving software quality [9]. From those review information, user requirements can be elicited continuously and, in a crowd-based fashion [11, 24].

The reviews of the mobile app users reflect a variety of topics, which majorly cover the perspectives of bug reports, feature requests, user experiences, solution proposal, information seeking and giving, and general ratings [17, 19, 21]. For such purpose, natural language processing (NLP), sentiment analysis (SA), and supervised learning are the common techniques used to classify user reviews [19, 21]. According to recent studies, the combination of NLP and SA techniques has high accuracy in detecting useful sentences [21]. These techniques are also used in many other studies in terms of review opinion mining [7, 9, 15].

Many studies have contributed to the user opinion mining of mobile apps. Fu et al. [8] proposes WisCom, which can detect why the users like or dislike a particular app based on the users' reviews throughout time and provide insights regarding the users' concerns and preferences in the app market. Similar studies regarding mining information from app stores data focus on different perspectives of the issues, e.g., the correlations between ratings and download rank [14], ratings and API use [2], review classification and useful sentences detection [21]. On the other hand, Chen et al. [7] provides the AR-miner framework, which facilitates the informative reviews extraction, review grouping based on topics, review prioritization and informative reviews presentation with visualization. Guzman and Maalej [12] proposes an approach focusing on feature extraction and sentiment analysis, which facilitates the evaluation of individual app features. Similarly, Iacob and Harrison [15] also focuses on the feature requests extraction but via means of linguistic rules and LDA topic modeling.

Compared to the previously mentioned approaches in user review opinion mining, our method aims towards the similar topic detection and analysis concerning not only the trend in users' opinions in general but also the specific users' opinion changes at particular updates. Furthermore, we focus on the topic similarity of data segments, classified by sentiment analysis and supervised learning, which is different from the methods mentioned above. It enables the developers to acquire the overall perspective regarding the users' opinions throughout the mobile app evolution phase, but also the information regarding each individual update providing insights on the future updates.

## 5. Conclusion

In this paper[7], we propose a method for analyzing the mobile apps user reviews for a series of updates. After classifying the reviews before and after a particular update by

[7]The work was partially supported by the TEKES Finnish project Virpa D.

positive and negative sentiment, we extract the topics of each segment. By comparing the similarities of these extracted topics, we identify not only the positive and negative issues reflected by these reviews regarding particular updates, but also the overall trend in users' opinions regarding the app in general as well as regarding specific aspects. Overall, this study is an exploratory investigation on using user review opinion mining techniques in detecting update-specific issues and user opinion evolution. The future studies shall extend the use of this method by taking into account other factors, e.g., the different app categories, different platforms, etc. And different mining and analysis techniques will also be taken into account. Furthermore, a mobile app evolution recommend system can be developed based on this method to provided maintenance and evolution suggestions to the developers regarding the continuous user opinion mining result.

## References


[1] S. Baccianella, A. Esuli, and F. Sebastiani. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, 2010.

[2] G. Bavota, M. Linares-Vasquez, C. E. Bernal-Cardenas, M. Di Penta, R. Oliveto, and D. Poshyvanyk. The impact of api change-and fault-proneness on the user ratings of android apps. *IEEE Trans. on Soft. Engin.*, 41(4):384–407, 2015.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[4] M. M. Bradley and P. J. Lang. Affective norms for english words (anew): Instruction manual and affective ratings. Technical report, Citeseer, 1999.

[5] P. Fafalios, V. Iosifidis, K. Stefanidis and E. Ntoutsi. Multi-aspect Entity-centric Analysis of Big Social Media Archives. In *Proceedings of TPDL*, 2017.

[6] E. Cambria, R. Speer, C. Havasi, and A. Hussain. Senticnet: A publicly available semantic resource for opinion mining. In *AAAI fall symposium: commonsense knowledge*, volume 10, 2010.

[7] N. Chen, J. Lin, S. C. Hoi, X. Xiao, and B. Zhang. Ar-miner: mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th International Conference on Software Engineering*, 2014.

[8] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh. Why people hate your app: Making sense of user feedback in a mobile app store. In *KDD*, 2013.

[9] L. V. Galvis Carreño and K. Winbladh. Analysis of user comments: an approach for software requirements evolution. In *ICSE*, 2013.

[10] C. H. E. Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *ICWSM*, 2014.

[11] E. C. Groen, J. Doerr, and S. Adam. Towards crowd-based requirements engineering a research preview. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, 2015.

[12] E. Guzman and W. Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *RE*, 2014.

[13] I. Ntoutsi, K. Stefanidis, K. Rausch and H. P. Kriegel. Strength Lies in Differences - Diversifying Friends for Recommendations through Subspace Clustering. In *Proceedings of CIKM*, 2014.

[14] M. Harman, Y. Jia, and Y. Zhang. App store mining and analysis: Msr for app stores. In *MSR*, 2012.

[15] C. Iacob and R. Harrison. Retrieving and analyzing mobile apps feature requests from online reviews. In *Mining Software Repositories*, 2013.

[16] K. Stefanidis, H. Kondylakis, and G. Troullinou. On Recommending Evolution Measures: A Human-aware Approach. In *Proceedings of ICDE Workshops*, 2017.

[17] H. Khalid, E. Shihab, M. Nagappan, and A. E. Hassan. What do mobile app users complain about? *IEEE Software*, 32(3):70–77, 2015.

[18] K. Stefanidis, E. Pitoura and P. Vassiliadis. Managing Contextual Preferences. *Information Systems*, 36(8):1158–1180, 2011.

[19] W. Maalej and H. Nabil. Bug report, feature request, or simply praise? on automatically classifying app reviews. In *RE*, 2015.
[20] D. Pagano and W. Maalej. User feedback in the appstore: An empirical study. In *RE*, 2013.
[21] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall. How can i improve my app? classifying user reviews for software maintenance and evolution. In *ICSME*, 2015.
[22] Y. Roussakis, I. Chrysakis, K. Stefanidis, G. Flouris, and Y. Stavrakas. A Flexible Framework for Understanding the Dynamics of Evolving RDF Datasets. In *Proceedings of ISWC*, 2015.
[23] K. Schneider. Focusing spontaneous feedback to support system evolution. In *RE*, 2011.
[24] N. Seyff, F. Graf, and N. Maiden. Using mobile re tools to give end-users their own voice. In *RE*, 2010.
[25] M. Stevenson and Y. Wilks. Word sense disambiguation. *The Oxford Handbook of Comp. Linguistics*, pages 249–265, 2003.
[26] Y. Roussakis, I. Chrysakis, K. Stefanidis, and G. Flouris. D2V: A Tool for Defining, Detecting and Visualizing Changes on the Data Web. In *Proceedings of ISWC*, 2015.

# PUBLICATION
# IV

**A statistical analysis of Steam user profiles towards personalized gamification**
X. Li, C. Lu, J. Peltonen and Z. Zhang

*International GamiFIN Conference (GamiFIN)*, 2019

# A statistical analysis of Steam user profiles towards personalized gamification

Xiaozhou Li, Chien Lu, Jaakko Peltonen, and Zheying Zhang

Tampere University
Kalevantie 4, 33100, Tampere, Finland
{xiaozhou.li, chien.lu, jaakko.peltonen, zheying.zhang}@tuni.fi

**Abstract.** Gamification is widely used as motivational design towards enhancing the engagement and performance of its users. Many commonly adopted game design elements have been verified to be effective in various domains. However, the designs of such elements in the majority of the target systems are similar. Due to inevitable differences between users, gamification systems can perform more effectively when users are provided with differently and personally designed features according to their preferences. Many studies have suggested such requirements towards personalizing gamified systems based on the users' preferences, with categorizing gamification users and identifying their preferences as the initial step. This study proposes a preliminary analysis of the factors that categorize user preference in a game community, based on the user profiles data of the Steam platform. It shall not only facilitate understanding of players' preferences in a game community but also lay the groundwork for the potential personalized gamification design.

**Keywords:** Gamification · Exploratory Factor Analysis · Steam · User Profile · Preference · Personalized Gamification.



## 1 Introduction

Gamification, commonly defined as the use of game design elements for non-game contexts [12], has been widely adopted as motivational design to support users motivation enhancement and performance improvement. Many game design elements, e.g., badges/achievements, points, leaderboard, progress, story, etc., have been adopted in various service domains and proven effective in many studies [14]. However, the majority of the gamification systems provide very limited alteration towards different users but adopt the one-size-for-all design approach instead [32]. Such rigid gameful designs are to a certain extent ineffective in persuading the users into positive behaviors. Many studies have shown that different users are likely to be motivated by different game elements and persuasive strategies [31, 32, 40]. Therefore, it is critical to understand different users' preferences when providing them the personalized gameful experiences.

The studies on the users' types and preferences regarding gamification systems are based on the similar studies on game players. A seminal study on the player types for multi-user dungeon (MUD) games is Bartle's player typology [2]. Meanwhile, a number of studies also contribute to extending the user typology framework by focusing

on psychographic and behavioral aspects [15]. Even though the direct connection is not addressed, such studies on player typology do facilitate the understanding of users preference of play style and their motivations of playing [15]. On the other hand, a gamification-specific user typology framework is developed by Marczewski [26], who proposes six gamification user types based on intrinsic or extrinsic motivational affordances [36] and their different degrees for the users. Furthermore, based on this particular framework, a 24-item survey response scale is presented to score users' preferences regarding the six different types of motivation toward a gameful system, which can therefore identify a users type and describe his/her preferences [42].

Despite the uniform well-defined player types and gamification user types, such a 'clear-cut' categorization approach can be questioned as a player may not belong to a certain type strictly [15, 21]. In addition, limitations of using survey data towards such categorization have also been recognized [42]. In this study, we focus on users of the Steam platform and their community-related behaviors presented on their profile pages. The users' Steam profiles provide various information, including the games they have, the game achievements, item trading, friends, groups, reviews, screenshots, profile customization options, and so on. The objective nature and large volume of such data shall has the potential to yield enhanced characterizations of users and their diferences. Herein, based on factor analysis of large user profile data, we identify the factors that characterize the differences between Steam users. Instead of a strict categorization of players, the study aims to answer what are the factors that distinguish Steam users from one another and determine their preferences, as well as how such distinguishing factors can be applied to facilitate personalized gamification design.

The paper is organized as follows. Section 2 introduces previous studies on game players and gamification user typologies and on analysis of the Steam platform and user data. Section 3 introduces our data collection and analysis methods, Sections 4 and 5 present results and discussion Section 6 concludes.

## 2 Related Work

### 2.1 Player Types and Gamification User Types

The aim of segmentation in marketing is to identify different customer groups so that they are served with products and services that match their unique needs. Studies on player types also serve this purpose. The majority of the prevailingly cited studies focus on the player segmentation in terms of the behavioral and psychographic attributes instead of geographic or demographic ones [15]; our focus is similar, since our Steam profiles did not contain demographic/geographic attributes and we focused on the available profile information reflecting player behavior. When available, our modeling principle could accommodate demographic/geographic attributes as covariates.

Bartle's seminal player typology — Achiever, Explorer, Socializer and Killer — is based on the things people enjoy about MUD in either an action or interaction dimension towards either players or the game world [2]. It is also criticized for being dichotomous and too simplifying, as well as focusing on only one game genre instead of a broad range [3, 15, 42]. Extending Bartle's typology model, many studies have proposed similar typology models for online game players with specialized focuses [43, 45]. Many

other studies present different ways of categorizing players based on their various motivation and behaviors when not fixating on online games [21, 39]. Such player typology models provide ways to detect the difference in players and their preference regarding motivations and behaviors in general. On the other hand, many studies also focus more specifically on players' preferences regarding game design elements [11, 19].

The studies on gamification user types also adapt the results from the player typology studies. Such studies are mostly supported by the research on behavior motivations and personalities [29,36]. Regarding the user typology in the gamification domain, Marczewskis gamification user type model is the most cited work [26]. Motivated by the intrinsic and extrinsic motivational factors of the users, which is defined by the Self Determination Theory (SDT) [35], Marczewski categorizes the users of gamification services into six types, including socializers, achievers, philanthropists, free spirits, players, and disrupters. Other studies also attempt to provide adapted typology frameworks regarding specific domains [1,44]. Meanwhile, adapting Marczewski's gamification user types model, Tondello et al. present and validate a standard scale to determine users' preference towards gamification systems regarding different motivation types [42]. Based on that, their subsequent works contribute to suggesting gameful design elements regarding user preferences, personalizing persuasive strategies, and creating a recommender system model for personalized gamification [32, 40, 41]. However, mentioned as their limitation, the data are self-reporting and subject heavily to participants' personal understanding of survey statements and preferences towards diverse game elements. Thus, relevant objective data with a larger sample volume can address such limitation and can also yield alternative results.

### 2.2 The Steam Platform and Users

Steam, a popular digital game distribution platforms, has drawn attention from the academia. Becker et al. analyze the role of games and groups in the Steam community and present the evolution of its network over time [5]. O'Neill et al. also investigate the Steam community but focus on the gamers' behaviors, in terms of their social connectivity, playtime, game ownership, genre affinity, and monetary expenditure [30], whereas Blackburn et al. focus more specifically on the cheating behavior [7]. Many other studies also investigate the various perspectives of players' behaviors on the Steam platform. For example, Sifa et al. investigate the players' engagement and cross-game behavior by analyzing their different playtime frequency distributions [37,38]. Baumann et al. focus on "hardcore" gamers' behavioral categories based on their Steam profiles [4]. Lim and Harrell examine players' social identity and the relation between their profile maintaining behaviors and their social network size [22]. Meanwhile, other scholars also study the other perspectives of Steam, such as, recommender systems for its content [6], early access mechanism [24], game updating strategies [23], game reviews [25], and so on. However, research on characterizing players based on their Steam profile data towards analyzing players' preference to different game design elements is still limited.

## 3 Method

### 3.1 Data Collection

A web crawler based on the Beautiful Soup Python module was created to collect data from public user profiles. The data collection proceeded in a "snowball" manner. The crawler started from one user's Steam profile URL which was selected at random from the top 10 Steam user leaderboard, and crawled the list of the user's friends profile URL. Iteratively, the list of users was grown via crawling the friends of each of the existing users on the list and appending the results to the end of the list. Although guaranteeing an unbiased sample from such a huge base is difficult and our gathered dataset is necessarily small, it can still achieve a good representativity. Duplicated profile URLs, as well as private ones from which no valid data can be obtained, were eliminated. To reduce crawling time while achieving reasonable coverage, only profile URLs were crawled, and from the initial data pool of 2561387 unique user profile URLs, we collected the profile information on a random subset of the URLs which includes 60267 users. The crawled features include *Levels*, *Showcases*, *Badges*, *Number of Games*, *Screenshots*, *Workshop Items*, *Videos*, *Reviews*, *Guides*, *Artworks*, *Groups*, *Friends*, *Items Owned*, *Trades Made*, *Market Transactions*, *Achievements*, *Perfect Games*, *Game Completion Rate*, and four binary *profile customization* related variables: *Avatar*, *Status*, *Background*, and *Favorite Badge* customization (customized or not). To summarize the binary variables per user, we define an aggregate value called *Profile Customization* whose value is the percent of 'customized' values: for example, if a particular user customized three of the four items mentioned above, his/her *Profile Customization* score will be assigned as 0.75. In addition, each user's active time span was also collected based on the time when the user last logged off and the time when the user created the account, using the SteamAPI. To take the user activity into account, we further computed the duration the profile had existed using the above-mentioned information and utilized it to normalize the profile variables, by simply dividing each variable by the profile duration.

### 3.2 Exploratory Factor Analysis

To uncover the underlying structures of the Steam user profiles, an exploratory factor analysis (EFA, [13]) is conducted. It enables us to reduce the complexity of the data, explain the observations with a smaller set of latent factors and discover the relations between variables. Unlike clustering which discovers groups of players, EFA discovers underlying axes characterizing players and their differences. In game culture studies, EFA has been widely used especially in studies related to user/player types and user motivations (e.g. [42, 43]). Extracted EFA factors can also be a basis for analysis such as clustering (player segmentation) or prediction in follow-up work; we focus on discovering underlying axes of variation in Steam user profiles through EFA and their applications in gamification.

One common issue in EFA is how to decide the number of factors. In this paper, the parallel analysis (PA) introduced by Horn [18] is adopted to solve the problem. It has been widely used and has given good results in recent research works (e.g. [33,

34]). Several comparative studies (e.g. [8, 46]) have shown that it is an effective way to determine the number of factors.

**Table 1.** Result of Parallel Analysis

| Factor | Observed Eigenvalue | Simulated Eigenvalue |
|---|---|---|
| 1 | 3.104 | 1.031 |
| 2 | 2.744 | 1.025 |
| 3 | 1.650 | 1.021 |
| 4 | 1.382 | 1.018 |
| 5 | 1.167 | 1.015 |
| 6 | 1.130 | 1.011 |
| 7 | 1.073 | 1.008 |
| 8 | 1.027 | 1.006 |
| 9 | 0.916 | 1.003 |

In PA, the Monte Carlo simulation technique is employed to simulate random samples consisting of uncorrelated variables that *parallel* the number of samples and variables in the observed data. From each such simulation, eigenvalues of the correlation matrix of the simulated data are extracted, and the eigenvalues are, as suggested in the original paper [18], averaged across several simulations. The eigenvalues extracted from the correlation matrix of the observed data, ordered by magnitude, are then compared to the average simulated eigenvalues, also ordered by magnitude. The decision criteria is that the factors with observed eigenvalues higher than the corresponding simulated eigenvalues are considered significant. Hereby, we conduct the parallel analysis task with 5000 simulations to determine the number of factors.

To simplify interpretation of the factor analysis result, the *varimax* rotation technique [20] which maximizes the variance of the each factor loading is employed. Results with an alternative rotation approach *promax* [17] were similar.

## 4 Result

### 4.1 Factor Analysis

The result of the parallel analysis is shown in Table 1. Based on the mentioned criteria, the turning point can be found easily by examining the differences between observed eigenvalues and simulated eigenvalues. Since the simulated eigenvalue becomes greater than the observed eigenvalue in the 9th factor (1.003 and 0.916 respectively), the first 8 factors are retained. The corresponding factor loadings can be found in Table 2. A cross-loading of the variable Profile.Customization was found on Factor 1 and 7, we further computed the Cronbach's alpha [9] on those two factors to evaluate their internal consistency and the values are found acceptable (0.87 and 0.71 respectively).

### 4.2 Factors Interpretation

Based on the result of EFA, we interpret each of the eight factors and summarize each of the unique preference attributes of Steam users.

**Table 2.** Loadings of the Extracted Factors

| Variable | Factor 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Level | **0.641** | -0.005 | 0.004 | -0.002 | 0.008 | -0.013 | -0.263 | 0.002 |
| Showcases | 0.026 | 0.107 | 0.065 | **0.828** | 0.162 | 0.180 | 0.028 | 0.067 |
| Badges | **0.954** | 0.033 | 0.004 | 0.010 | 0.006 | 0.043 | 0.016 | 0.004 |
| Games | 0.019 | **0.511** | 0.020 | 0.016 | 0.108 | 0.365 | 0.030 | 0.088 |
| Screenshots | -0.000 | 0.118 | 0.332 | 0.046 | 0.344 | 0.039 | 0.022 | **0.490** |
| Workshop.Items | 0.007 | -0.045 | 0.042 | 0.127 | **0.789** | -0.027 | 0.003 | -0.082 |
| Videos | 0.002 | -0.030 | -0.066 | 0.046 | -0.074 | -0.022 | -0.003 | **0.901** |
| Reviews | 0.002 | 0.232 | 0.039 | 0.044 | **0.769** | 0.039 | 0.018 | 0.113 |
| Guides | 0.002 | 0.024 | **0.879** | -0.031 | -0.090 | -0.003 | -0.001 | -0.002 |
| Artwork | 0.004 | -0.010 | **0.836** | 0.101 | 0.192 | 0.006 | 0.018 | 0.030 |
| Groups | 0.078 | 0.017 | 0.020 | 0.031 | 0.026 | 0.008 | **0.951** | 0.009 |
| Friends | **0.947** | 0.002 | 0.004 | 0.043 | 0.007 | 0.014 | 0.202 | 0.001 |
| Items.Owned | 0.004 | 0.048 | 0.005 | 0.049 | -0.004 | **0.733** | 0.006 | -0.022 |
| Trades.Made | -0.003 | -0.142 | -0.002 | 0.281 | -0.063 | **0.551** | 0.003 | -0.061 |
| Market.Transactions | 0.017 | 0.116 | 0.001 | -0.063 | 0.044 | **0.645** | -0.007 | 0.049 |
| Achievements | 0.005 | **0.865** | 0.014 | 0.125 | 0.014 | -0.010 | -0.001 | -0.011 |
| Perfect.Games | 0.003 | **0.847** | 0.006 | 0.210 | 0.105 | -0.045 | -0.002 | -0.017 |
| Game.Completion.Rate | 0.008 | 0.274 | 0.013 | **0.852** | 0.054 | -0.004 | 0.003 | 0.021 |
| Profile.Customization | **0.808** | -0.007 | -0.008 | -0.019 | -0.015 | -0.016 | **0.553** | -0.007 |

**Factor 1: Elite (Level, Badge, Friends, and Profile Customization)** Factor 1 indicates the users' tendency to become the elite of the Steam community. The *elite* users focus on their social comparison advantages over the others by enhancing their quantifiable social scores, such as, levels, badges, and friends numbers. According to Steam's unique mechanism, the users can upgrade their levels and earn more badges without the requirements of exerting more effort in actual gameplay. Therefore, the elite users tend to value their social achievement more than experiences in gameplay. In addition, they also prefer profile customization in order to present their unique social identity.

**Factor 2: Achiever (Games, Achievement, and Perfect Games)** Users' tendency in Factor 2 indicates their preference towards mastering the games. They focus on completing games thoroughly and obtaining as many in-game achievements as possible. They also tend to enlarge their game collection whenever possible. Compared to the *elite* users, the *achiever* users prefer to put their effort in games and less in social.

**Factor 3: Provider (Guides and Artworks)** Users with high attribute in Factor 3 love to provide facilitation to the others with gameplay guides and self-created unique game-related arts. Different from *elite* and *achiever* users who focus on their social presence or achievement, the *provider* users tend to be more altruistic and care about other users and their game playing.

**Factor 4: Completer (Showcases and Game Completion Rate)** Similar to the *achiever* users, the *completer* users also focus on gameplay but less on achievements. They prefer to finish the games that they start but have less intention of pursuing the full achievement by investing extra amount of hours. Meanwhile, they like to show their possessions, e.g., showcases, as much as possible, but put less effort on organizing compared with the *elite* users.

**Factor 5: Improver (Workshop Items and Reviews)** Users with high value on Factor 5 focus on game improvement. They make efforts to add unique experiences to games via workshop items and reviews. These encourage developers to improve the games and publish better games in the future. Similar to *provider* users, they are also altruistic but focus more on game quality.

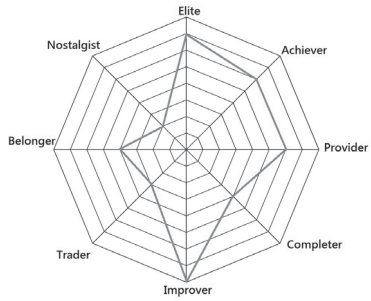


**Fig. 1.** An Example of User Preference Attributes Radar Chart

**Factor 6: Trader (Item Owned, Trades Made, and Market Transaction)** The *trader* users do not pay much attention to either games or social, but to buying and selling game related virtual items instead. According to Steam's mechanism, users neither have to own or play games to obtain items nor have to become friends with others or join groups to make trades. Thus, *trader* users tend to make the community a business playground, buying low and selling high.

**Factor 7: Belonger (Groups and Profile Customization)** Similar to the *elite* users, the *belonger* users also tend to focus more on social interaction than gameplay, when the difference is that the belonger users prefer the feeling of relatedness and belonging, rather than social comparison. Belonging to social groups is always their first priority. Having a proper customized profile is thus also necessary to fit them in the groups.

**Factor 8: Nostalgist (Screenshots and Videos)** Users with high *nostalgist* attribute have the tendency of restoring their gameplay memories by taking screenshots and recording videos. They also share their gameplay memories with others in the activity timeline, so that other players can enjoy the unique scenes and compare to their own gameplay too. Meanwhile, the "thumbs up" and appreciation from the others is their reward.

It is worth noting that the eight factors aim to explore the various attributes of Steam users instead of arbitrarily categorizing each user into a single type. Generally, each individual user shall contain certain scores in all given attributes while the attribute value distribution of different users shall differ. Meanwhile, each user may also contain high or low score in multiple attributes simultaneously. By reducing the variable dimensions to one for each attribute and normalizing the value, each individual user shall have a radar chart illustrating his/her salient attributes. Fig. 1 shows an example of a user who possesses a salient attribute of *improver* and is creative with workshop items and also loves to contribute in improving games by giving reviews. Meanwhile, this particular

user also possesses relevantly salient attributes of *elite*, *achiever*, and *provider*. It indicates that the user also favors gaining levels, badges, and achievements, and providing guides and artworks to the community.

**Table 3.** An Example Mapping between Preference Attributes and Motivation Types

| Attributes | Steam Variables | Motivation Types [10, 36] | Gameful Elements [40] |
|---|---|---|---|
| Elite | Level | Mastery | Progression |
| | Badges | Mastery | Incentive |
| | Friends | Relatedness | Socialization |
| | Profile Customization | Autonomy | Customization |
| Achiever | Games | Mastery | Progression |
| | Achievements | Mastery | Incentive |
| | Perfect Games | Mastery | Incentive |
| Provider | Guides | Mastery, Purpose | Altruism |
| | Artwork | Autonomy | Altruism |
| Completer | Showcases | Autonomy, Mastery | Customization |
| | Game Completion Rate | Mastery | Progression |
| Improver | Workshop Items | Autonomy, Purpose | Altruism |
| | Reviews | Autonomy, Purpose | Altruism |
| Trader | Items Owned | Mastery | Incentive |
| | Trades Made | Relatedness | Socialization |
| | Market Transactions | Relatedness | Socialization |
| Belonger | Groups | Relatedness | Socialization |
| | Profile Customization | Autonomy | Customization |
| Nostalgist | Screenshots | Autonomy, Relatedness | Socialization |
| | Videos | Autonomy, Relatedness | Socialization |

To apply such a preference framework in gamification design, based on the variables each attribute is related to, we could find connections between attributes and the established intrinsic motivation types or other similar gamification design models or frameworks. With different player motivation and design elements frameworks, the application towards personalized gamification design could differ. Table 3 is an example of connecting the obtained preference attributes with the SDT motivation types [10, 36] and the gameful design elements categories [40]. Ideally, each Steam variable can be mapped to a certain type of motivation and a particular gameful design element category. Subsequently, the motivation that drives the corresponding preference attributes and the related gameful design element set can be decided and weighted (e.g., based on relatedness of the variables to the attributes). However, such presumption of connecting attributes, motivation types, and design elements can be subjective, when the motivation of each user towards each individual Steam variable is unknown and hard to be dichotomized. For example, 'Level' is likely to be driven by the motivation of mastery, when, on the other hand, particularly in Steam, higher level means that the user will have more badges and showcases to customize. Therefore, the 'Level' variable

is driven by the motivation of autonomy, to some extent. Furthermore, a quantifiable value of 'Level', together with 'Badges' and 'Profile Customization', can be also seen as the tendency towards social comparison. Such equivocality shall be addressed with potential ordering or voting schemes.

## 5 Discussion

Compared with Lim and Harrell's study on players' social identity [22], we cover more perspectives of Steam users' social behaviors in the gamer community by extending the data collection to more features. However, different from Sifa et al.'s work [38] our data covers only the Steam users' profile information and not users' in-game behaviors. Thus, with the current dataset, mapping from the obtained user preferences towards the gameful design elements regarding heavily in-game behaviors, such as, immersion or risk/reward, is not possible [40]. Furthermore, based on the goal of this study to study users' preference regarding gamification design, the data limits generalization towards all gamification users instead of only gamers. Despite the above limitations, the data (similar to other product-oriented social media profiles, e.g. Amazon profiles) can be seen as more generalized rather than focusing on gamers from specific games or genres. Compared with previous studies on gamification user types [40,42], such data collected from user profiles can be more objective than self-reported survey data.

This study presents a data-driven approach to investigating users' preferences towards game design elements. The resulting axes of variation among players can be inspected and used in gamification. In future work the results can also be used as a basis for categorization of players; data-driven approaches [16] can improve efficiency and representativeness compared to manually designed categories. One follow-up direction is to build a collaborative filtering recommender system based on similarity of users' preference towards various game design elements, allowing a personalized gamification design based on the recommendation for each user [41]. Another future direction is to validate the user preference framework with empirical analysis. For example, the user preference scale of Tondello et al. [42] can be adopted as a reference, with Steam users as participants. Furthermore, the data volume can be enlarged with more users, e.g., by crawling from multiple seed users; our data could further be combined with additional data regarding, e.g., players' in-game behaviors, preference on game genres, and reviews on games. After validation, the proposed user preference framework can be applied to future data-driven player studies. Together with previous gamification design methods [27], the framework will facilitate gamification design and provides an efficient way to address key issues in the user analysis phase [28].

## 6 Conclusion

We presented an exploratory way of analyzing user presences towards game design elements using Steam user profile data. Using EFA, eight factors/attributes are gained, the value of which can be used to define each individual user's preference regarding behaviors in the Steam community. Together with the connection between such behaviors and the underlying motivation types and gameful design elements, each user's preference

regarding gamification systems can be also perceived. Due to the quantifiable and objective nature of the data, such estimation of the users' preference can be more precise. It will contribute to the future work of personalized gamification design and creation of recommender systems for personalized gamification in a data-driven manner.

*Acknowledgments.* This research was supported by the Academy of Finland project Centre of Excellence in Game Culture Studies (CoE-GameCult, 312395).

## References

1. Barata, G., Gama, S., Jorge, J.A., Gonçalves, D.J.: Relating gaming habits with student performance in a gamified learning experience. In: Proceedings of the first ACM SIGCHI annual symposium on Computer-human interaction in play. pp. 17–25. ACM (2014)
2. Bartle, R.: Hearts, clubs, diamonds, spades: Players who suit muds. Journal of MUD research **1**(1), 19 (1996)
3. Bateman, C., Lowenhaupt, R., Nacke, L.: Player typology in theory and practice. In: DiGRA Conference (2011)
4. Baumann, F., Emmert, D., Baumgartl, H., Buettner, R.: Hardcore gamer profiling: Results from an unsupervised learning approach to playing behavior on the steam platform. Procedia Computer Science **126**, 1289–1297 (2018)
5. Becker, R., Chernihov, Y., Shavitt, Y., Zilberman, N.: An analysis of the steam community network evolution. In: Electrical & Electronics Engineers in Israel (IEEEI), 2012 IEEE 27th Convention of. pp. 1–5. IEEE (2012)
6. Bertens, P., Guitart, A., Chen, P.P., Periáñez, Á.: A machine-learning item recommendation system for video games. arXiv preprint arXiv:1806.04900 (2018)
7. Blackburn, J., Simha, R., Kourtellis, N., Zuo, X., Long, C., Ripeanu, M., Skvoretz, J., Iamnitchi, A.: Cheaters in the steam community gaming social network. arXiv preprint arXiv:1112.4915 (2011)
8. Crawford, A.V., Green, S.B., Levy, R., Lo, W.J., Scott, L., Svetina, D., Thompson, M.S.: Evaluation of parallel analysis methods for determining the number of factors. Educational and Psychological Measurement **70**(6), 885–901 (2010)
9. Cronbach, L.J.: Coefficient alpha and the internal structure of tests. psychometrika **16**(3), 297–334 (1951)
10. Deci, E.L., Eghrari, H., Patrick, B.C., Leone, D.R.: Facilitating internalization: The self-determination theory perspective. Journal of personality **62**(1), 119–142 (1994)
11. Denisova, A., Cairns, P.: First person vs. third person perspective in digital games: Do player preferences affect immersion? In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. pp. 145–148. ACM (2015)
12. Deterding, S., Dixon, D., Khaled, R., Nacke, L.: From game design elements to gamefulness: defining gamification. In: Proceedings of the 15th international academic MindTrek conference. pp. 9–15. ACM (2011)
13. Hair, J.F., Black, W.C., Babin, B.J., Anderson, R.E., Tatham, R.L., et al.: Multivariate data analysis (vol. 6) (2006)
14. Hamari, J., Koivisto, J., Sarsa, H.: Does gamification work?–a literature review of empirical studies on gamification. In: 2014 47th Hawaii international conference on system sciences (HICSS). pp. 3025–3034. IEEE (2014)
15. Hamari, J., Tuunanen, J.: Player types: A meta-synthesis. Transactions of the Digital Games Research Association **1**(2), 29 (2014)
16. Han, J., Pei, J., Kamber, M.: Data mining: concepts and techniques. Elsevier (2011)

17. Hendrickson, A.E., White, P.O.: Promax: A quick method for rotation to oblique simple structure. British journal of statistical psychology **17**(1), 65–70 (1964)
18. Horn, J.L.: A rationale and test for the number of factors in factor analysis. Psychometrika **30**(2), 179–185 (1965)
19. Hsu, S.H., Kao, C.H., Wu, M.C.: Factors influencing player preferences for heroic roles in role-playing games. CyberPsychology & Behavior **10**(2), 293–295 (2006)
20. Kaiser, H.F.: The varimax criterion for analytic rotation in factor analysis. Psychometrika **23**(3), 187–200 (1958)
21. Kallio, K.P., Mäyrä, F., Kaipainen, K.: At least nine ways to play: Approaching gamer mentalities. Games and Culture **6**(4), 327–353 (2011)
22. Lim, C.U., Harrell, D.F.: Developing social identity models of players from game telemetry data. In: AIIDE (2014)
23. Lin, D., Bezemer, C.P., Hassan, A.E.: Studying the urgent updates of popular games on the steam platform. Empirical Software Engineering **22**(4), 2095–2126 (2017)
24. Lin, D., Bezemer, C.P., Hassan, A.E.: An empirical study of early access games on the steam platform. Empirical Software Engineering **23**(2), 771–799 (2018)
25. Lin, D., Bezemer, C.P., Zou, Y., Hassan, A.E.: An empirical study of game reviews on the steam platform. Empirical Software Engineering pp. 1–38 (2018)
26. Marczewski, A.: Even ninja monkeys like to play: Gamification, game thinking & motivational design. Gamified UK (2015)
27. Mora, A., Riera, D., Gonzalez, C., Arnedo-Moreno, J.: A literature review of gamification design frameworks. In: 2015 7th International Conference on Games and Virtual Worlds for Serious Applications (VS-Games). pp. 1–8. IEEE (2015)
28. Morschheuser, B., Hassan, L., Werder, K., Hamari, J.: How to design gamification? a method for engineering gamified software. Information and Software Technology **95**, 219–237 (2018)
29. Nacke, L.E., Bateman, C., Mandryk, R.L.: Brainhex: preliminary results from a neurobiological gamer typology survey. In: International Conference on Entertainment Computing. pp. 288–293. Springer (2011)
30. O'Neill, M., Vaziripour, E., Wu, J., Zappala, D.: Condensing steam: Distilling the diversity of gamer behavior. In: Proceedings of the 2016 Internet Measurement Conference. pp. 81–95. ACM (2016)
31. Orji, R., Nacke, L.E., Di Marco, C.: Towards personality-driven persuasive health games and gamified systems. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. pp. 1015–1027. ACM (2017)
32. Orji, R., Tondello, G.F., Nacke, L.E.: Personalizing persuasive strategies in gameful systems to gamification user types. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. p. 435. ACM (2018)
33. Pontes, H.M., Griffiths, M.D.: Measuring dsm-5 internet gaming disorder: Development and validation of a short psychometric scale. Computers in Human Behavior **45**, 137–143 (2015)
34. Reilly, A., Eaves, R.C.: Factor analysis of the minnesota infant development inventory based on a hispanic migrant population. Educational and psychological measurement **60**(2), 271–285 (2000)
35. Ryan, R.M., Deci, E.L.: Intrinsic and extrinsic motivations: Classic definitions and new directions. Contemporary educational psychology **25**(1), 54–67 (2000)
36. Ryan, R.M., Deci, E.L.: Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. American psychologist **55**(1), 68 (2000)
37. Sifa, R., Bauckhage, C., Drachen, A.: The playtime principle: Large-scale cross-games interest modeling. In: CIG. pp. 1–8 (2014)
38. Sifa, R., Drachen, A., Bauckhage, C.: Large-scale cross-game player behavior analysis on steam. Borderlands **2**, 46–378 (2015)

39. Stewart, B.: Personality and play styles: A unified model. Gamasutra, September **1** (2011)
40. Tondello, G.F., Mora, A., Nacke, L.E.: Elements of gameful design emerging from user preferences. In: Proceedings of the Annual Symposium on Computer-Human Interaction in Play. pp. 129–142. ACM (2017)
41. Tondello, G.F., Orji, R., Nacke, L.E.: Recommender systems for personalized gamification. In: Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization. pp. 425–430. ACM (2017)
42. Tondello, G.F., Wehbe, R.R., Diamond, L., Busch, M., Marczewski, A., Nacke, L.E.: The gamification user types hexad scale. In: Proceedings of the 2016 annual symposium on computer-human interaction in play. pp. 229–243. ACM (2016)
43. Tseng, F.C.: Segmenting online gamers by motivation. Expert Systems with Applications **38**(6), 7693–7697 (2011)
44. Xu, Y., Poole, E.S., Miller, A.D., Eiriksdottir, E., Kestranek, D., Catrambone, R., Mynatt, E.D.: This is not a one-horse race: understanding player types in multiplayer pervasive health games for youth. In: Proceedings of the ACM 2012 conference on computer supported cooperative work. pp. 843–852. ACM (2012)
45. Yee, N.: Motivations for play in online games. CyberPsychology & behavior **9**(6), 772–775 (2006)
46. Zwick, W.R., Velicer, W.F.: Comparison of five rules for determining the number of components to retain. Psychological bulletin **99**(3), 432 (1986)

# PUBLICATION

# V

**A Sentiment-Statistical Approach for Identifying Problematic Mobile App Updates Based on User Reviews**

X. Li, B. Zhang, Z. Zhang and K. Stefanidis

*Information* 11.3 (2020), 152

information

MDPI

Article

# A Sentiment-Statistical Approach for Identifying Problematic Mobile App Updates Based on User Reviews

**Xiaozhou Li *, Boyang Zhang *, Zheying Zhang * and Kostas Stefanidis ***

Faculty of Information Technology and Communication Sciences (ITC), Tampere University, 33520 Tampere, Finland

* Correspondence: xiaozhou.li@tuni.fi (X.L.); boyang.zhang@tuni.fi (B.Z.); zheying.zhang@tuni.fi (Z.Z.); konstantinos.stefanidis@tuni.fi (K.S.)

Received: 22 January 2020; Accepted: 10 March 2020; Published: 12 March 2020


**Abstract:** Mobile applications (apps) on IOS and Android devices are mostly maintained and updated via Apple Appstore and Google Play, respectively, where the users are allowed to provide reviews regarding their satisfaction towards particular apps. Despite the importance of user reviews towards mobile app maintenance and evolution, it is time-consuming and ineffective to dissect each individual negative review. In addition, due to the different app update strategies, it is uncertain that each update can be accepted well by the users. This study aims to provide an approach to detect the particular days during the mobile app maintenance phase when the negative reviews require developers' attention. Furthermore, the method shall facilitate the mapping of the identified abnormal days towards the updates that result in such negativity in reviews. The method's purpose is to enable app developers to respond swiftly to significant flaws reflected by user reviews in order to prevent user churns.

**Keywords:** mobile app; sentiment analysis; maintenance; update; user review; exponential power distribution; Word2Vec

## 1. Introduction

Contemporary mobile applications (apps) play an increasingly important role in people's daily lives, which are directly influenced by the apps' quality. Via the dynamic way of distribution supported by the platforms e.g., Apple Appstore and Google Play, the release periods of mobile apps are largely shortened compared to those of traditional software products [1]. Due to the fiercely competitive mobile app market, the developers are obliged to constantly update their app products by fixing bugs, improving interfaces, adapting to system updates, and providing new features, in order to maintain the quality of the apps and to increase user retention and satisfaction [1,2]. In the constant updating process, understanding end users' needs and requests is important and requires enormous effort. There are various ways of getting access to users' feedback for a particular app. Besides the dedicated in-app user feedback collecting tools, the online app stores contain a mechanism to allow users to provide text reviews and rating scores, which enhances their importance as a stakeholder [3].

The reviews given by end users are helpful towards improving the app quality in general through its maintenance and evolution with proper analysis, despite the proportion of informative reviews is around one third [4,5]. Many studies have proposed approaches and techniques facilitating the analysis of mobile app user reviews [5–11]. Therein, sentiment analysis and topic modeling are often applied for the analysis with various focuses, e.g., on detecting review characteristics [12], on identifying review inconsistency and user concerns [11], on review classification [7,8], on the controversiality about the sentiment towards specific entities [13], or even on comparing them with users ratings [14].

Furthermore, many studies also contribute to applying other methods of review analysis in order to tackle other software maintenance and evolution related issues, e.g., release planning [15,16], change requests localizing and recommendation [17], as well as software suggestions [18].

On the other hand, due to the dynamic distribution mechanism and constant market changes, mobile apps are continuously updated with a rapid pace, though such an update mechanism implementation is risky for potential user dissatisfaction [19]. Despite most users being happy with the apps with frequent updates but hesitate to install them, worrying about potential hazards [1]. Most of the previous studies focus on eliciting users' general opinion from the reviews regarding a particular mobile app as a whole, while also lacking support on the detection of updates, which may adversely affect user experience and satisfaction. We call such updates problematic updates. Releasing continuously problematic updates can stop users from using the particular app again, which in turn impacts the app's exposure opportunities on the app stores. Identifying the problematic updates early on and understanding the causes of user dissatisfaction can help mobile app providers predict potential problems before releasing a new update which ensures the success of the app in its life cycle. Li et al. propose an approach to analyzing the topic and sentiment changes before and after a particular update, but has a lack of support for identifying the particular time and the potential update that requires attention [10]. Xia et al. provide a way to predict mobile app crashing updates based on commit data analysis, but lack of facilitation towards identifying generally problematic updates [20].

In this paper, we propose a sentiment-statistical approach of identifying the problematic mobile app updates based on user review analysis, by specifically answering the following research questions:

1. How to identify the collective dissatisfaction of users based on their reviews?
2. How to verify it is the recent update that results in the users' dissatisfaction?

The reminder of the article is organized as follows. Section 2 introduces the related studies regarding similar topics. Section 3 presents our method with details. Section 4 presents a case study, applying the proposed method on retrieved mobile app review data. Section 5 further discusses the relevant issues, as well as the limitation of the study and future works. Section 6 concludes the article with a summary of our contributions.

## 2. Related Work

Regarding the use of data mining techniques for user reviews analysis towards mobile app quality, as well as practice regarding maintenance and evolution, many previous studies focus on aspects, like the helpfulness and informativeness of the reviews, feature extraction and review classification [21]. For example, towards informative review identification, Chen et al. adopt the expectation maximization for the naive Bayes method to classify informative and non-informative reviews and topic modeling methods to group informative reviews [5]. Gao et al. adopt the Info-rate index to analyze the informative rate of the reviews and track the dynamics of top-rated reviews without manual labeling [22]. Chandy and Gu propose a method to identify spam reviews with the baseline decision tree model and latent class model [23]. Towards feature extraction, Vu et al. propose a keyword-based framework for semi-automated review analysis facilitating the extraction of review keywords and the mapping to the related negative reviews [24]. Fu et al. adopt statistical analysis and topic modeling method to discover inconsistency in reviews and identify the major concerns of the users marked by extracted keywords [11]. Many other studies also use topic modeling methods, such as, LDA and ASUM, to extract major concerns of users from the reviews and the related features [4,6,25]. Furthermore, despite qualitative and exploratory methods also being used for the classification of user reviews, specifically the complaint types of the users [2,26], many scholars still adopt natural language processing, topic modeling and sentiment analysis techniques for such purpose [8,10,27].

Together with the opinion mining of user reviews, many studies focus on the continuous maintenance of mobile apps via update analysis and release planning. Villarroel et al., propose the

CLAP method to categorize and cluster user reviews based on extracted features and to prioritize and recommend review clusters towards the planning of the subsequent updates [15]. Ciurumelea et al., propose the user request referencer prototype to not only classify reviews but also map the reviews according to source code files that can be modified to address the issues within [16]. On the other hand, towards the analysis of mobile app updates, Wang et al. use a k-means clustering algorithm to identify seven mobile app update patterns based on the feature intensity trend between two neighboring updates, reflecting the common update behaviors towards acting on user reviews [28]. Li et al, present a method with topic modeling and sentiment analysis to analyze the changes of user opinions through continuous app updates [29]. Overall, the previous works on mobile app reviews opinion mining largely focus on issues related to detection, and provide corresponding strategies for future updates. This study, on the other hand, aims to facilitate the identification of particular updates that result in statistically abnormal amount of negative reviews.

## 3. Method

In this work, we propose an approach aiming to identify the periods of time when the user reviews of a particular mobile app reflect noticeable amount of negativity. Such identification of the opinion changes shall also be correlated with the changes of the topics in the according review periods. This approach also aims to identify the particular update that is most likely connected to such sentiment and main topic changes by comparing the similarity between the keywords of the update content and those of the reviews from the period of such changes. In general, this approach encompasses two key steps: (1) identifying the abnormal days of changes in the sentiment of user reviews, and (2) identifying the corresponding problematic update that is most likely connected to such abnormality. The outcomes of these two steps, respectively, answer the research questions mentioned above.

### *3.1. Sentiment Change Distribution*

In order to identify the period of time during the maintenance and evolution of a particular mobile app, where the overall user review sentiment of that period is abnormal, we shall firstly be able to differentiate such abnormality from normal review sentiment. By observing the rating percentage of changes of a number of popular mobile apps from the last 90 days (from 13 June 2020 to 10 September 2020), we find that the rating percentages are largely stable, despite the varied proportion number from app to app. For example, such stability in the rating proportions through days can be easily observed in Figure 1, which shows the rating percentage changes for 90 days for mobile app Whatsapp (Obtained from https://www.appannie.com).

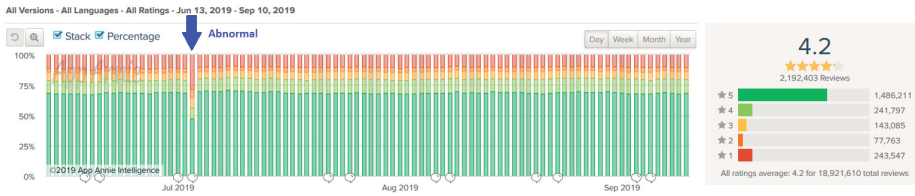


**Figure 1.** The rating percentage changes of Whatsapp from 13 June 2020 to 10 September 2020.

Figure 1 shows an abnormal rating proportion that is noticed on the 3rd July 2019, when the percentage of 1-star ratings rose from an average 10% up to 28.4%, while that of 5-star ratings dropped from 70% down to 47.4%. Such sudden rating proportion changes on that particular day can be assumed to be reflecting the changes in the quality of the app via the recent update. On the other hand, user ratings are somehow inconsistent towards the review content provided [11], when the reviews are considered more accurately reflecting the user's opinion. In addition, based on the results from the 1-year data of reviews and ratings from the five mobile apps given next (3,793,125 reviews), we find

the correlation between the daily average sentiment and ratings is high (with Pearson's $r = 0.984$). Due to such high correlation between ratings and review sentiment, the phenomenon of sudden changes in rating proportion caused by updates can also be observed by the changes in the collective review sentiment.

Accordingly, when considering such phenomena of daily average review sentiment being stable, we can then propose a hypothesis that having the review data divided by a fixed time period (e.g., by day), the changes of the average sentiments of the obtained review divisions along the timeline are normally distributed. To test the hypothesis, we use the Kolmogorov–Smirnov test (K-S test) [30] for the fitness of negative sentiment changes of the previously mentioned data of five mobile apps to normal distribution. However, the hypothesis does not stand with $p$ taking value equal to 0.000001 (less than 0.05). Furthermore, in order to find the best fitting distribution model, we apply the K-S test for 86 other different distribution models (https://docs.scipy.org/doc/scipy/reference/stats.html) finding that our data fits best to generalized normal distribution (version 1. Exponential power distribution, shown in Figure 2) [31] with a $p$ value of 0.968 ($\mu = 0.0002$, $\alpha = 0.0227$, $\beta = 1.0444$).

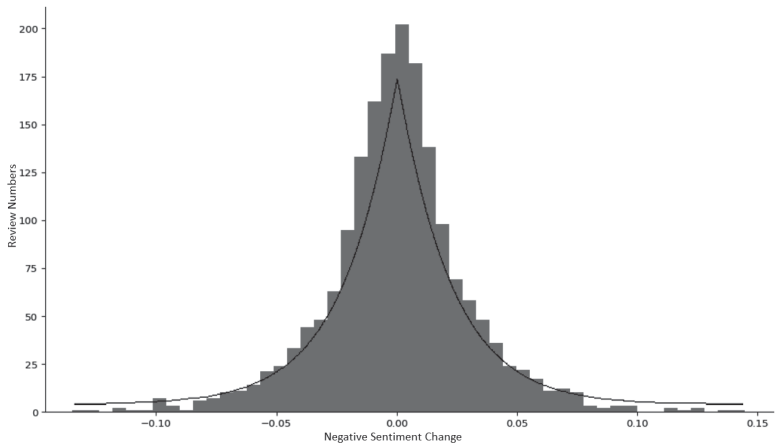


**Figure 2.** The distribution of the review data negative sentiment change.

### *3.2. Identify Abnormal Sentiment Changes*

The overall review sentiment of a particular day can be defined in various ways, such as the average sentiment score of the reviews on each day or the rate of certain type of sentiment (i.e., positive or negative). For review sentiment analysis, due to the inconsistency between the sentiment expressed by users and the actual quality, it is highly likely the majority of moderate negative reviews can be neutralized by the minority positive reviews with exaggerated expression. Furthermore, as the method is to identify the abnormal day when a significant number of negative reviews occur, we hereby define the sentiment change between two days as the increase or decrease rate of negative review numbers, where the strength of sentiment is chosen to be ignored. Thus, considering respectively $n_i$ and $n_{i+1}$ negative reviews found in the $N_i$ reviews on the day $d_i$ and $N_{i+1}$ reviews on the next day $d_{i+1}$, the review sentiment changes between day $d_i$ and day $d_{i+1}$, denoted as $as_i$, is calculated as $as_i = (n_{i+1}/N_{i+1}) - (n_i/N_i)$.

Based on the hypothesis proposed previously, if the variables of daily sentiment changes fit normal distribution, we can then consider the samples that lie within the band around the mean in a normal distribution with a width of six standard deviations as normal (i.e., the three sigma rule [32]). However, for the non-normal distributions, such inference towards the percentage of normal values may vary. Based on the previous exponential power distribution model for the review sentiment change data, we simulate 1,000,000 samples having such distribution with the obtained

parameters (i.e., $\mu = 0.0002$, $\alpha = 0.0227$, $\beta = 1.0444$). The simulation is from the R package *gnorm* (https://cran.r-project.org/web/packages/gnorm/index.html). The calculated number of samples lying in the band around the mean with $\pm 3\sigma$ width is 986,424, indicating 98.6% of the values shall be considered normal. Thus, the abnormal sentiment changes of the reviews are the ones lying outside the band. On the other hand, considering Chebyshev's inequality [33], no more than $1/k^2$ of the distribution's values can be more than $k$ standard deviations away from the mean. Specifically, regarding the simulation data, the abnormal values are the $1 - 1/k^2 = 0.986$ of the distribution, that is, around $k = 8$ standard deviations away from the mean. Therefore, regarding our dataset, it is reasonable to consider the days when the sentiment change values lying out the 98.6% band around the mean as abnormal.

### *3.3. Identify Problematic Updates*

Despite the changes in review, sentiment scores can be used to identify the potential problems in the mobile app quality, it is possible that such changes result from the general quality deterioration instead of from particular problematic updates [34]. A way of finding the connection between the identified abnormal review sentiment changes and the corresponding problematic updates is to detect the similarity between the content of the reviews that causing the changes and that of the updates. Towards such purpose, we adopt the *Word2Vec* tool [35].

The Word2Vec tool uses vectors to represent words with efficient and continuous bag-of-words and skip-gram schemes [36]. Each word from the text corpus generated from the text data is transformed into an individual vector. The vocabulary model is constructed by training the text data, which uses a log-linear classifier to predict words occurring within a certain range to either side of the word and learn the word representation. Simply put, for a particular word in the corpus, it is less likely related to the words occurring frequently far away from it. Thus, the similarity between this word and the words far away weighs less.

In order to investigate the similarity between the content of reviews and that of the update, we firstly train the Word2Vec model with the textual review data of a particular app. After identifying the days when the sentiment changes are considered abnormal, we summarize the negative reviews of that day using the term frequency–inverse document frequency (TF-IDF), which reflects the importance of a word to a document (i.e., review text) in a collection of corpus (i.e, collection of reviews) [37]. Thereafter, the similarity between the content of a particular update and that of the negative reviews of the identified abnormal days can be measured by averaging the similarities between the words with high TF-IDF value of the abnormal-day negative reviews and the keywords extracted from the update description text. Ideally, when the main cause of a particular abnormal day is verified, it is most likely the nearest update before the abnormal day that results in such abnormality. Otherwise, further investigation into the details of the reviews is required to verify the causes.

### *3.4. Algorithm*

Algorithm 1 offers an overview of the proposed approach. Specifically, our approach consists of three main steps:

1. Calculate the parameters of the exponential power distribution model of the daily review sentiment changes.
2. Identify the abnormal sentiment change days based on the distribution model, if any.
3. Check whether it is the nearest previous update that is problematic and causing such detected abnormality in user review sentiment change by comparing the similarity between the negative reviews of the abnormal days and the update description texts.

**Algorithm 1:** Algorithm of identifying abnormal days and problematic updates.

**Data:** Set of Reviews within a Fixed Period
**Result:** Identified Problematic Update if Abnormal Sentiment Change Detected
INITIALIZATION;
$R \leftarrow$ set of reviews;
**for** ***each*** $r_i \in R$ **do**
  $s_i$ ($\in S$) $\leftarrow$ getSentimentScore($r_i$)
**end**
$D \leftarrow$ set of days, where R is obtained;
**for** ***each*** $\{r_x, r_{x+1}, \ldots r_{x+m}\}$ *obtained in* $d_i \in D$ **do**
  Let $as_i$ be the average review sentiment for $d_i$;
  $as_i \leftarrow$ len([$s$ in $\{s_x, s_{x+1}, \ldots s_{x+m}\}$ if $s$ is negative])/m
**end**
**for** ***each*** $d_i \in D$ **do**
  Let $sc_i$ be the sentiment change between $d_i$ and $d_{i+1}$;
  $sc_i \leftarrow (as_{i+1} - as_i)$
**end**
Let $X$ be the continuous random variable of sentiment changes;
Then, as $X \sim EPD(\mu, \alpha, \beta)$, calculate $\mu, \alpha$, and, $\beta$;
Let $AD$ be the set of abnormal sentiment change time spans;
$AD \leftarrow$ [for $sc_i$ in $X$ if $sc_i > \mu + 3\sigma$];
**if** $AD$ *exist* **then**
  Let $U$ be the set of updates released within the fixed period;
  $Ut \leftarrow$ [the update content of $u$ for $u$ in $U$];
  **for** ***each*** $ad_i \in AD$ **do**
    $F \leftarrow$ sorted(getTF-IDFList([reviews in $d_{i+1}$]), reverse=True)[:150];
    $K \leftarrow$ [getKeywords($ut$) for $ut$ in $Ut$];
    **if** *the* $u_i$ *with the max([Similarity(k, F) for k in K]) is the nearest previous update* **then**
      return $u_i$;
    **else**
      $ad_i$ is not caused by update;
    **end**
  **end**
**else**
  no abnormal days detected
**end**

Let $R$ be a set of user reviews for a particular mobile app $A$ within a defined time period, where each individual review $r_i \in R$ is tagged with a specific time point. Meanwhile, let $U$ also be the set of updates released by the developers of app $A$ within the same time period, where each update $u_i \in U$ is also released at a particular time point. Thus, if any abnormal days are detected, for each identified day with abnormal sentiment change during the period, $ad_i$, we can verify whether it is the nearest previous update that is problematic and causing $ad_i$. Herein, the similarity between the negative reviews and the previous update is calculated by the average of the Word2Vec similarity values of each update description keyword and each of the top TF-IDF review keywords. The according TF-IDF are also taken into account as the weight of the similarity values. To be noted, we hereby select 150 keywords with the highest TF-IDF score to ease the calculation cost of the algorithm, while maintaining its accuracy. If based on the comparison of similarities, the nearest previous update is not identified as problematic, the negativity in user reviews can be caused by other issues, which requires further investigation.

## 4. Case Study

This case study is to validate the usefulness of the proposed method in identifying the abnormal days from user reviews and the problematic updates during the mobile app maintenance. We collect the review data from five popular mobile apps and apply our method. The result shows that the problematic updates can be identified when using this method.

### *4.1. Data Preprocessing*

Preprocessing on the acquired raw review data is required before experimenting with the proposed method. We hereby apply the following steps to clean the data into usable.

**Filtering non-English reviews.** We screen out the non-English review sentences using Langdetect [38], a convenient language detecting package for Python language. Langdetect identifies the language of a particular sentence using the sentence as a whole instead of individual words within. Hence, due to the nature of user reviews, the sentences with misspelled words, slurs and abbreviations, used often in social media, will not be filtered out.

**Separating Long Reviews into Sentences.** Despite mobile app reviews being shorter, in general, compared with reviews on other platforms, particular reviews still contain multiple sentences, each of which might convey different meanings and sentiments. Therefore, we use the sentence tokenizer feature from the NLTK [39] python package to obtain the sentence set of each individual review item.

**Calculating the Sentiments.** Herein, we use the Valence Aware Dictionary for sEntiment Reasoning (VADER) approach [40] to calculate the sentiment score of each review sentence. VADER is a commonly used sentiment analysis tool due to its classification accuracy on sentiment towards positive, negative and neutral classes, which is even higher than individual human raters in the social media domain. According to Hutto and Gilbert's experiment results [40], the F1 score of VADER on social media text (i.e., short text with informal language) is 0.96. Due to the unique trait of mobile app reviews being short and informal compared to other reviews types (e.g., movie reviews), we expect such sentiment analysis being as accurate as that on social media text. To further verify the accuracy of the VADER approach on mobile app reviews, we calculate the precision, recall and F1 score. Shown in Table 1, the overall accuracy of VADER on mobile app review sentences is 0.819. When considering multi-sentences reviews, we calculate the sentiment score of each review as the mean of scores of each sentence in the review. Shown in Table 1, the accuracy of overall accuracy of VADER on mobile app reviews is 0.842.

**Table 1.** Sentiment score accuracy testing.

| App | Review-Level | | | Sentence-Level | | |
|---|---|---|---|---|---|---|
| | **Precision** | **Recall** | **F1 Score** | **Precision** | **Recall** | **F1 Score** |
| Imo | 0.826 | 0.786 | 0.804 | 0.822 | 0.777 | 0.797 |
| Hangouts | 0.839 | 0.803 | 0.819 | 0.802 | 0.780 | 0.790 |
| Messenger | 0.883 | 0.835 | 0.855 | 0.869 | 0.815 | 0.836 |
| Skype | 0.874 | 0.828 | 0.844 | 0.828 | 0.790 | 0.795 |
| Whatsapp | 0.851 | 0.814 | 0.830 | 0.833 | 0.787 | 0.805 |
| Overall | 0.868 | 0.823 | 0.842 | 0.850 | 0.800 | 0.819 |

In addition, compared to the other popular sentiment analysis approaches, such as SenticNet [41], SentiWordNet [42], Affective Norms for English Words [43] and Word-Sense Disambiguation [44], its overall classification accuracy on product reviews from Amazon, movie reviews and editorials from NYTimes also prevails. Furthermore, VADER is easy to import and use as it is integrated in the NLTK package.

**Filtering Stopwords and Lemmatization.** Every review sentence shall contain words that are used often but carry less meaning, i.e., the stopwords. The stopwords must be removed in order to obtain meaningful term frequency results. The adopted stopwords set also includes the app-specific

terms with high occurence, like *'app'*, *'application'* and *'whatsapp'*, as well as the common typos of other stopwords, like *'dont'*, *'whasapp'* and *'appp'*. In addition, the acquired review sentences are also transformed into lower cases and lemmatized before the calculation of term frequency so that the same term with different cases shall be seen as one. We hereby use the stopword set in NLTK corpus package to screen the stopwords from review sentences, and the *WordNetLemmatizer* from NLTK stem package to lemmatize words.

**Selecting Nouns and Verbs.** Nouns and verbs are two major word types we consider in term frequency analysis. Adjectives and adverbs are filtered because the term frequency analysis is to gain an insight into the issues in reviews with a negative sentiment score. The adjectives and adverbs which mainly contribute to the sentiment score have been considered in sentiment analysis, and require no redundant covering in the term frequency analysis. We use the RegexpTokenizer from NLTK and the pos_tags of the tokenized words to identify and filter words that are not nouns or verbs.

### *4.2. Data Description*

In this study, we use the user reviews of five instant messenger mobile apps from Android platform, namely Imo, Hangouts, Messenger, Skype and Whatsapp. We collect for each app the reviews from 1 September 2016 to 31 August 2017, eliminate the non-English reviews and tokenize each into sentences. The numbers of reviews and the obtained English reviews and numbers of sentences for each app are shown in Table 2, while the number of review sentences on each day is shown in Figure 3-left. Together with the user reviews, we also collect the release date information regarding the updates within the given period. The numbers of updates counted for each app are also shown in Table 2.

**Table 2.** Application (app) review statistics.

| App Name | Reviews | English Reviews | Sentences | Updates |
|---|---|---|---|---|
| Imo | 202,870 | 86,194 | 100,838 | 84 |
| Hangouts | 122,622 | 68,535 | 10,1704 | 43 |
| Messenger | 1,654,360 | 886,643 | 1,185,368 | 105 |
| Skype | 153,128 | 105,875 | 189,995 | 76 |
| Whatsapp | 1,660,145 | 851,662 | 109,8583 | 49 |
| total | 3,793,125 | 1,998,909 | 2,676,488 | 357 |

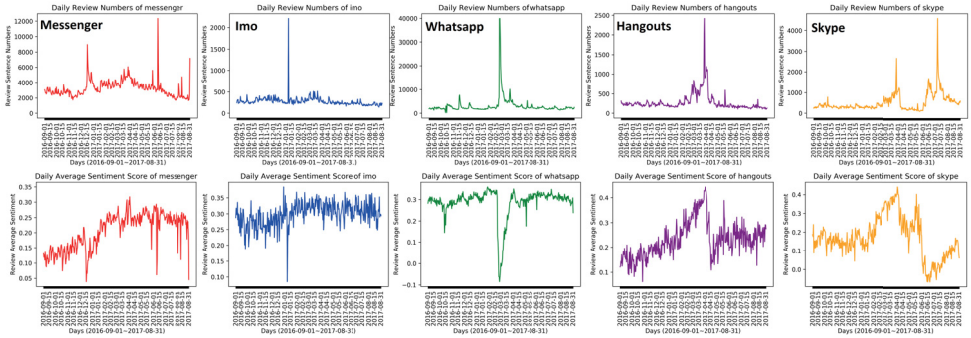


**Figure 3.** Number of reviews and average sentiment score by day (from 2016-09-01 to 2017-09-01).

Thereafter, for each of the 2,676,488 review sentences, we calculate its sentiment score using the VADER sentiment algorithm. The sentiment scores range from 1.0 (i.e., very positive) to −1.0 (i.e., very negative). Furthermore, we calculate the daily average sentiment score for each selected app and plot the changes in Figure 3. Such average sentiment score is calculated as follows. Provided $m$ reviews are given on the day $d_i$ with the sentiment score of each review $r_i$ within is calculated and denoted as

$s_i$ ($i = 1, 2, \ldots, m$), the average sentiment score on $d_i$ is calculated as $(\sum_{i=1}^{m} s_i)/m$. From the sentiment changes, we can observe that the majority of the daily sentiment range from 0.1 to 0.4. It indicates that for these five apps, the overall sentiment within the given period is still positive despite such changes. In addition, we find that the daily average rating of each app is highly correlated with the daily average sentiment with an average Pearson $R$ score of 0.93. It indicates the sentiment of the reviews can be used to reflect the users' evaluation to their general fondness of the apps.

### *4.3. Results*

Herein, we apply the previously presented method, that is, identifying abnormal days based on the distribution of negative sentiment changes and matching such abnormality to a particular update, on each review dataset of the given five mobile applications. By doing so, we aim to investigate whether such a method can be used towards obtaining meaningful results.

#### 4.3.1. Identify Abnormal Days

For the 1-year review sentiment data of the given five mobile apps, we firstly calculated the sentiment changes of each day as the negative review proportion change. Considering all 1840 obtained daily sentiment change values as the continuous random variable, we detect the most likely distribution model that fits the values. By doing so, we find that the best fitting distribution model is the exponential power distribution (EPD), with a $p$ value of 0.968. Furthermore, the parameters for the obtained EPD model are $\mu = 0.0002$, $\alpha = 0.0227$ and $\beta = 1.0444$.

Based on the obtained distribution parameters, we calculate the confidential intervals ($\mu \pm 3\sigma$) for each set of sentiment change values for each mobile app (shown in Table 3). Based on the obtained confidential intervals, we can identify the abnormal days of each mobile app, where the sentiment changes are greater than $\mu + 3\sigma$.

**Table 3.** Confidential intervals and identified abnormal days.

| App Name | CI | Abnormal Days (Year-Month-Day) |
|---|---|---|
| Imo | (−0.082, 0.083) | ['13 December 2016', '7 January 2017', '3 August 2017'] |
| Hangouts | (−0.098, 0.099) | [ ] |
| Messenger | (−0.060, 0.061) | ['16 December 2016', '9 June 2017', '3 August 2017', '31 August 2017'] |
| Skype | (−0.095, 0.096) | ['4 September 2016', '21 May 2017', '25 May 2017'] |
| Whatsapp | (−0.053, 0.052) | ['12 October 2016', '15 October 2016', '21 February 2017', '23 February 2017', '24 February 2017', '3 May 2017'] |

By observing the daily sentiment changes, we can easily map the obtained abnormal days results with the significant sentiment changes from the evolution chart. Figure 4-left shows the review sentiment changes of Whatsapp. The x-axis represents the consecutive dates, while the y-axis represents the proportion of positive, neutral and negative reviews (shown in green, blue and red curves, respectively). The six abnormal days can be observed by the obvious rise in negative sentiment and fall of positive sentiment. Furthermore, such abnormal days can also be validated by the changes of top frequent words of each day. By comparing the Jaccard similarity of the top frequent words of each day, we can also find that the top frequent words of abnormal days are largely different from those of other days. The similarity values of the top 10 frequent words between each day for Whatsapp are shown in Figure 4-right, where the dark color indicates the low similarity value. Therein, we can observe that the identified abnormal days contain different top frequent words from the other days, when the reviews of the other days largely share common top frequent words.

In addition, we can also observe that all the days on which top frequent words are different from those of the others are not identified as abnormal days. For example, from 2017-02-20 to 2017-03-31, the daily average review sentiments of Whatsapp are continuously more negative than usual, despite the rising from 2017-02-27. Accordingly, we can observe that the similarities among the top frequent

words during that period are high, but also low towards other days. Thus, an implication can be made that the negative effect of the identified abnormal days lasts for the whole period regarding similar issues.

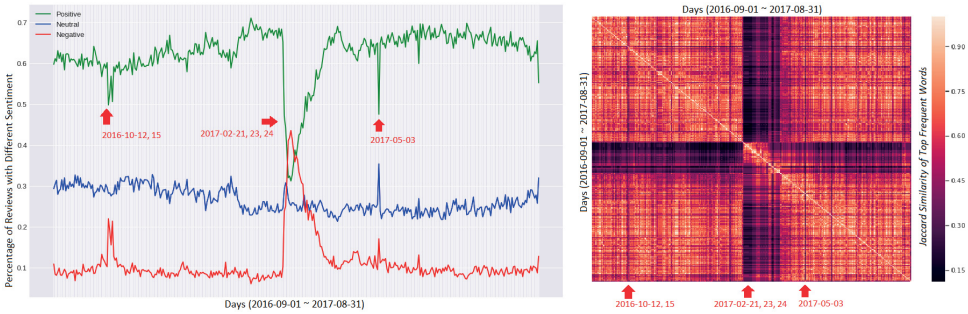


**Figure 4.** The identified abnormal days of Whatsapp as an example.

4.3.2. Identify Problematic Updates

Based on the obtained abnormal days of the given mobile apps, we continue to investigate the particular updates that result in such abnormality in review sentiment, if any. For such purpose, we retrieve the update description text of each update during the review period for the mobile apps. Herein, the update description texts are required to specifically describe, to a certain extent, the update content for each particular new version. Therefore, we only select Whatsapp and Skype for this experiment, due to the fact that the update description texts of IMO and Messenger are vague and identical throughout the period, and no abnormal days are identified for Hangouts. Furthermore, we take into account only the major updates, which are identified as the first updates whose description text is different from that of the previous one(s). For example, shown in Figure 5, Version 2017.06.16 is identified as a major update when the other ones are identified as minor ones.

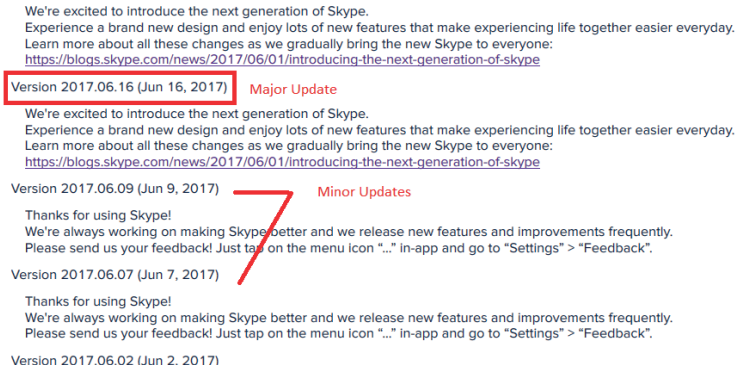


**Figure 5.** Examples of major and minor updates.

In order to identify the cause of each identified abnormal day, we compare the similarity between the negative review content of the identified abnormal days and the description text of the major updates. As mentioned in the prior section, we select only nouns and verbs from both the update description and reviews. Furthermore, for the review texts, we select only the words with high TF-IDF scores, which represent the main content of the text.

Figure 6 shows the similarity of the negative reviews of each identified abnormal day and the descriptions of the nine major updates. Therein, the nearest previous update of each abnormal day is marked red. As shown in the figure, for the identified abnormal days of Whatsapp: 2017-02-21,

2017-02-23, 2017-02-24 and 2017-05-03, the similarities of the negative reviews and the descriptions of their nearest previous updates are significantly high. By observing the review texts of these four dates, we locate 20017 negative review sentences out of 32,922 containing the keyword "update" or "version", which shows the connection between the review negativity of the abnormal days and the most recent updates. However, for the abnormal days 2016-10-12 and 2016-10-15, the similarities are not as significant as with the four ones shown in the first two charts of Figure 6. We investigate closely on the update description text of Version 2016.10.11, stating "*Now you can draw or add text and emojis to photos and videos you capture within WhatsApp...New emoji. And sending a single emoji will now appear larger in chats*". As we find "emoji" being the core feature updated in this particular version, 46 out of 105 negative review sentences on these two days contain the word "update", "version" or "emoji".

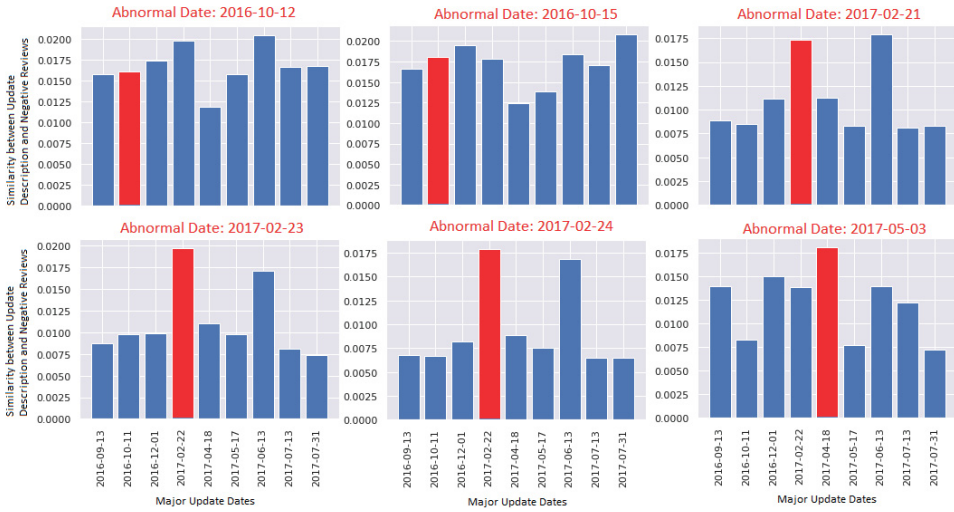


**Figure 6.** Similarity between update description and negative reviews of abnormal days for Whatsapp.

The result for Skype is shown in Figure 7, which is not significant compared to that for Whatsapp. The negative reviews of the identified abnormal days, 2017-05-21 and 2017-05-25, cannot be matched to their nearest previous update by similarities to the description text. Abnormal day 2016-09-04 is ignored here, as it occurs before the first update retrieved within the review data period. By further investigating the review texts of the two abnormal days, we find only 4 out of 70 review sentences contain word "update" or "version". The review texts are mostly describing quality related issues with scattered topics, which indicates that it is not the particular update that cause the negativity in reviews.

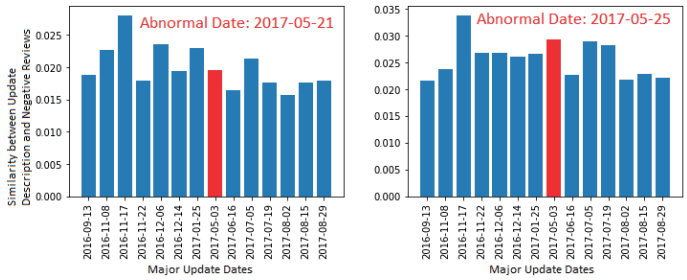


**Figure 7.** Similarity between update description and negative reviews of abnormal days for Skype.

To summarize, for a particular mobile app, the proposed method can be used to identify specific time period, e.g., days, during which the negative sentiment of the reviews is considered abnormal compared to a much larger set of reviews within a much longer period. It is also possible to identify the particular update, mostly the nearest one before an identified abnormal day, that causes such abnormality in review sentiment (e.g., the problematic Version 2017.02.22 of Whatsapp). However, when not the nearest update is identified with the highest similarity between negative reviews and the update description texts, it is likely that the negative reviews of the abnormal days are regarding more general and scattered issues, instead of a particular update (e.g., the result of Skype). The vague description of the update content and the limited review numbers can be the factors that result in various unpredictable outcomes with the method.

## 5. Discussion

Compared to the previous studies on mobile app review analysis [21], this study focuses on finding the negative user reviews in a particular day that matter the most to the developers in terms of sentiment changes, and investigating the connection between these reviews and the potential updates that cause the negativity. It is a light-weight method that eases the effort of the continuous analysis of the increasing amount of user reviews that also contain large volume of non-informative content. Regarding the mobile app maintenance practice, the method can facilitate the emergency-oriented maintenance model, which is usually adopted by the mobile apps with stable core functionalities [45]. Thus, our method can be considered more suitable for indie app developers [46] or a small development team to swiftly identify and fix problematic updates reflected by noticeable amount of negativity in reviews.

Taking into consideration the results of the case study above, the method performs well when the volume of review data is sufficient and the update description texts are detailed composed. Hence, one of the limitations of this study is the validation of the method towards particular mobile app cases, which receive only a limited amount of reviews. Furthermore, due to the adoption of distribution analysis, the method can also suffer from a "cold start" issue, that is, for a newly launched app product without sufficient review data, the method will fail to perform. Another critical limitation of this research is the threat to validity, as only the review data of five instant messenger mobile apps are taken into account. Thus, further verification of the method regarding its prerequisite on review data volume and update description texts is required. In addition, a further validation of the review sentiment change distribution model is also necessary, which can be done with a larger mobile app data repository. Accuracy of the sentiment analysis can, to a certain extent, influence the validity of the identified abnormal days. Thus, the adoption of such method on the analysis of other reviews, e.g., movie reviews, product reviews, etc., shall yield to such threat to validity, especially when the accuracy is relatively low and the volume of the data is limited.

In our future work, we will focus on addressing the limitations mentioned above and improving the method on the following aspects. Methods, such as aspect-based sentiment analysis [47] together with user review feature extraction [24], will largely enhance the effectiveness of the method in terms of the uneven user sentiment on various complaint types from mobile app users [2]. Similarly, topic model techniques can also be applied to extract app feature related topics, which will also enhance the usefulness of the method. Building on such results, the method will be able to prioritize the severity of topic-based or feature-based categorized issues and facilitate developers planning on updates. In addition, taking into account the different reviewing behaviors of app users, as well as their preferences on app types, can also provide insights on analyzing the helpfulness of the reviews given. On the other hand, a method for automatically evaluating the helpfulness of update description and extracting the features of the updates shall also be helpful towards reducing the human effort provided the number of apps selected for the future work being excessive.

## 6. Conclusions

This study proposes a sentiment-statistical approach for detecting abnormal days during mobile app maintenance. The core of the method is based on the analysis of review sentiment distribution, and the similarities between update descriptions and review texts. Specifically, critical conclusions can be made when the negative sentiment increases sharply in a particular time period. In addition, we use the same method to map abnormal days to potential updates that cause such days.

Our method aims to facilitate mobile app developers to identify the critical moment during mobile app evolution when the users' opinion towards the app product grows overwhelmingly negative very quickly, and checking whether such negativity is caused by the nearest update. The results of the case study show that the proposed method performs effectively, however, with the prerequisite of having sufficient volume of user review data and adequately detailed update description text.

**Author Contributions:** Conceptualization, X.L., B.Z., Z.Z. and K.S.; data curation, X.L.; formal analysis, X.L. and B.Z.; investigation, X.L., Z.Z. and K.S.; methodology, X.L.; software, X.L.; supervision, Z.Z. and K.S.; validation, X.L.; visualization, X.L.; writing—original draft, X.L. and B.Z.; writing—review and editing, X.L., Z.Z. and K.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Nayebi, M.; Adams, B.; Ruhe, G. Release Practices for Mobile Apps–What do Users and Developers Think? In Proceedings of the 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (Saner), Osaka, Japan, 14–18 March 2016; Volume 1, pp. 552–562.
2. Khalid, H.; Shihab, E.; Nagappan, M.; Hassan, A.E. What do mobile app users complain about? *IEEE Softw.* **2014**, *32*, 70–77.
3. Holzer, A.; Ondrus, J. Mobile application market: A developer's perspective. *Telemat. Inf.* **2011**, *28*, 22–31.
4. Galvis Carreño, L.V.; Winbladh, K. Analysis of user comments: an approach for software requirements evolution. In Proceedings of the 2013 International Conference on Software Engineering, San Francisco, CA, USA, 18–26 May 2013; pp. 582–591.
5. Chen, N.; Lin, J.; Hoi, S.C.; Xiao, X.; Zhang, B. AR-miner: mining informative reviews for developers from mobile app marketplace. In Proceedings of the 36th International Conference on Software Engineering, Hyderabad, India, 31 May 31–7 June 2014; pp. 767–778.
6. Guzman, E.; Maalej, W. How do users like this feature? a fine grained sentiment analysis of app reviews. In Proceedings of the 2014 IEEE 22nd international requirements engineering conference (RE), Karlskrona, Sweden, 25–29 August 2014; pp. 153–162.
7. Maalej, W.; Nabil, H. Bug report, feature request, or simply praise? on automatically classifying app reviews. In Proceedings of the 2015 IEEE 23rd international requirements engineering conference (RE), Ottawa, ON, Canada, 24–28 August 2015; pp. 116–125.
8. Panichella, S.; Di Sorbo, A.; Guzman, E.; Visaggio, C.A.; Canfora, G.; Gall, H.C. How can i improve my app? classifying user reviews for software maintenance and evolution. In Proceedings of the 2015 IEEE international conference on software maintenance and evolution (ICSME), Bremen, Germany, 27 September–3 October 2015; pp. 281–290.
9. McIlroy, S.; Ali, N.; Khalid, H.; Hassan, A.E. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empir. Softw. Eng.* **2016**, *21*, 1067–1106.
10. Li, X.; Zhang, Z.; Stefanidis, K. Sentiment-aware Analysis of Mobile Apps User Reviews Regarding Particular Updates. In Proceedings of the Thirteenth International Conference on Software Engineering Advances (ICSEA), Nice, France, 14–18 October 2018; p. 109.
11. Fu, B.; Lin, J.; Li, L.; Faloutsos, C.; Hong, J.; Sadeh, N. Why people hate your app: Making sense of user feedback in a mobile app store. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, IL, USA, 11–13 August 2013; pp. 1276–1284.
12. Zhang, L.; Hua, K.; Wang, H.; Qian, G.; Zhang, L. Sentiment analysis on reviews of mobile users. *Procedia Comput. Sci.* **2014**, *34*, 458–465.

13. Fafalios, P.; Iosifidis, V.; Stefanidis, K.; Ntoutsi, E. Multi-aspect Entity-Centric Analysis of Big Social Media Archives. In Proceedings of the Research and Advanced Technology for Digital Libraries—21st International Conference on Theory and Practice of Digital Libraries TPDL, Thessaloniki, Greece, 18–21 September 2017; pp. 261–273.
14. Stratigi, M.; Li, X.; Stefanidis, K.; Zhang, Z. Ratings vs. Reviews in Recommender Systems: A Case Study on the Amazon Movies Dataset. In *European Conference on Advances in Databases and Information Systems*; Springer: Cham, Switzerland, 2019.
15. Villarroel, L.; Bavota, G.; Russo, B.; Oliveto, R.; Di Penta, M. Release planning of mobile apps based on user reviews. In Proceedings of the 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE), Austin, TX, USA, 14–22 May 2016; pp. 14–24.
16. Ciurumelea, A.; Schaufelbühl, A.; Panichella, S.; Gall, H.C. Analyzing reviews and code of mobile apps for better release planning. In Proceedings of the 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), Klagenfurt, Austria, 20–24 February 2017; pp. 91–102.
17. Palomba, F.; Salza, P.; Ciurumelea, A.; Panichella, S.; Gall, H.; Ferrucci, F.; De Lucia, A. Recommending and localizing change requests for mobile apps based on user reviews. In Proceedings of the 39th International Conference on Software Engineering, Buenos Aires, Argentina, 20–28 May 2017; pp. 106–117.
18. Koskela, M.; Simola, I.; Stefanidis, K. Open Source Software Recommendations Using Github. In Proceedings of the Digital Libraries for Open Knowledge, 22nd International Conference on Theory and Practice of Digital Libraries TPDL, Porto, Portugal, 10–13 September 2018; pp. 279–285.
19. Möller, A.; Michahelles, F.; Diewald, S.; Roalter, L.; Kranz, M. Update behavior in app markets and security implications: A case study in google play. In Proceedings of the 14th International Conference on Human Computer Interaction with Mobile Devices and Services, San Francisco, CA, USA, 2012; pp. 3–6.
20. Xia, X.; Shihab, E.; Kamei, Y.; Lo, D.; Wang, X. Predicting crashing releases of mobile applications. In Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, Ciudad Real, Spain, 8–9 September 2016; p. 29.
21. Genc-Nayebi, N.; Abran, A. A systematic literature review: Opinion mining studies from mobile app store user reviews. *J. Syst. Softw.* **2017**, *125*, 207–219.
22. Gao, C.; Xu, H.; Hu, J.; Zhou, Y. Ar-tracker: Track the dynamics of mobile apps via user review mining. In Proceedings of the 2015 IEEE Symposium on Service-Oriented System Engineering, San Francisco, CA, USA, 30 March 30–3 April 2015; pp. 284–290.
23. Chandy, R.; Gu, H. Identifying spam in the iOS app store. In Proceedings of the 2nd Joint WICOW/AIRWeb Workshop on Web Quality, Lyon, France, 16 April 2012; pp. 56–59.
24. Vu, P.M.; Nguyen, T.T.; Pham, H.V.; Nguyen, T.T. Mining user opinions in mobile app reviews: A keyword-based approach (t). In Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), Lincoln, NA, USA, 9–13 November 2015; pp. 749–759.
25. Iacob, C.; Harrison, R. Retrieving and analyzing mobile apps feature requests from online reviews. In Proceedings of the 10th Working Conference on Mining Software Repositories, San Francisco, CA, USA, 18–19 May 2013; pp. 41–44.
26. Pagano, D.; Maalej, W. User feedback in the appstore: An empirical study. In Proceedings of the 2013 21st IEEE international requirements engineering conference (RE), Rio de Janeiro, Brazil, 15–19 July 2013; pp. 125–134.
27. Zimina, E.; Nummenmaa, J.; Järvelin, K.; Peltonen, J.; Stefanidis, K.; Hyyrö, H. GQA: Grammatical Question Answering for RDF Data. In Proceedings of the Semantic Web Challenges—5th SemWebEval Challenge at ESWC, Heraklion, Greece, 3–7 June 2018; pp. 82–97.
28. Wang, S.; Wang, Z.; Xu, X.; Sheng, Q.Z. App Update Patterns: How Developers Act on User Reviews in Mobile App Stores. In *International Conference on Service-Oriented Computing*; Springer: Berlin, Germany, 2017; pp. 125–141.
29. Li, X.; Zhang, Z.; Stefanidis, K. Mobile App Evolution Analysis Based on User Reviews. In Proceedings of the International Conference on Intelligent Software Methodologies, Tools, and Techniques, Granada, Spain, 26–28 September 2018; pp. 773–786.
30. Massey, F.J., Jr. The Kolmogorov-Smirnov test for goodness of fit. *J. Am. Stat. Assoc.* **1951**, *46*, 68–78.
31. Nadarajah, S. A generalized normal distribution. *J. Appl. Stat.* **2005**, *32*, 685–694.
32. Pukelsheim, F. The three sigma rule. *Am. Stat.* **1994**, *48*, 88–91.

33. Chebyshev, P.L. Des valeurs moyennes, Liouville's. *J. Math. Pures Appl.* **1867**, *12*, 177–184.
34. Hoon, L.; Vasa, R.; Schneider, J.G.; Grundy, J. *An Analysis of the Mobile App Review Landscape: Trends And Implications*; Faculty of Information and Communication Technologies, Swinburne University of Technology: Melbourne, Australia, 2013.
35. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*; The MIT Press: Cambridge, MA, USA, 2013; pp. 3111–3119.
36. Xue, B.; Fu, C.; Shaobin, Z. A study on sentiment computing and classification of sina weibo with word2vec. In Proceedings of the 2014 IEEE International Congress on Big Data, Anchorage, AK, USA, 27 June–2 July 2014; pp. 358–363.
37. Ramos, J. Using tf-idf to determine word relevance in document queries. In Proceedings of the First Instructional Conference on Machine Learning, Piscataway, NJ, USA, 3–8 December 2003; Volume 242, pp. 133–142.
38. Langdetect. Available online: https://pypi.python.org/pypi/langdetect (accessed on 30 September 2019).
39. NLTK. Available online: http://www.nltk.org (accessed on 30 September 2019).
40. Hutto, C.J.; Gilbert, E. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In Proceedings of the Eighth international AAAI conference on Weblogs and Social Media, Ann Arbor, MI, USA, 1–4 June 2014.
41. Cambria, E.; Speer, R.; Havasi, C.; Hussain, A. Senticnet: A publicly available semantic resource for opinion mining. In Proceedings of the AAAI Fall Symposium: Commonsense Knowledge, Arlington, VA, USA, 2010; Volume 10.
42. Baccianella, S.; Esuli, A.; Sebastiani, F. *Sentiwordnet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining*; Lrec: Baton Rouge, Louisiana, 2010; Volume 10, pp. 2200–2204.
43. Bradley, M.M.; Lang, P.J. *Affective Norms for English Words (ANEW): Instruction Manual and Affective Ratings*; Technical Report; University of Florida, The Center for Research in Psychophysiology: Gainesville, FL, USA, 1999.
44. Stevenson, M.; Wilks, Y. Word sense disambiguation. In *The Oxford Handbook of Computational Linguistics*; Oxford University Press: Oxford, UK, 2003; pp. 249–265.
45. Li, X.; Zhang, Z.; Nummenmaa, J. Models for mobile application maintenance based on update history. In Proceedings of the 2014 9th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE), Lisbon, Portugal, 28–30 April 2014; pp. 1–6.
46. Qiu, Y.; Gopal, A.; Hann, I.H. Logic pluralism in mobile platform ecosystems: A study of indie app developers on the iOS app store. *Inf. Syst. Res.* **2017**, *28*, 225–249.
47. Thet, T.T.; Na, J.C.; Khoo, C.S. Aspect-based sentiment analysis of movie reviews on discussion boards. *J. Inf. Sci.* **2010**, *36*, 823–848.


# PUBLICATION VI

**A preliminary network analysis on steam game tags: another way of understanding game genres**

X. Li and B. Zhang

*Proceedings of the 23rd International Conference on Academic Mindtrek*, 2020, 65–73

# A Preliminary Network Analysis on Steam Game Tags: Another Way of Understanding Game Genres

XIAOZHOU LI, Tampere University
BOYANG ZHANG, Tampere University

Video game genre classification has long been a focusing perspective in game studies domain. Despite the commonly acknowledged usefulness of genre classification, scholars in the game studies domain are yet to reach consensus on the game genre classification. On the other hand, Steam, a popular video game distribution platform, adopts the user-generated tag feature enabling players to describe and annotate video games based on their own understanding of genres. Despite the concern of the quality, the user-generated tags (game tags) provide an opportunity towards an alternative way of understanding video game genres based on the players' collective intelligence. Hence, in this study, we construct a network of game tags based on the co-occurrence of tags in games on Steam platform and analyze the structure of the network via centrality analysis and community detection. Such analysis shall provide an intuitive presentation on the distribution and connections of the game tags, which furthermore suggests a potential way of understanding the important tags that are commonly adopted and the main genres of video games.

CCS Concepts: • **Applied computing** → **Computer games**; • **General and reference** → Document types; • **Human-centered computing** → *Social network analysis.*

Additional Key Words and Phrases: Video Game, Genre, Game Tag, Network, Modularity, Centrality, Community Detection, Steam

## 1 INTRODUCTION

In various subjects, the usage of genre classification has become a normative tool of information managing, it helps users to identify, locate, and retrieve items of interest within mutually exclusive divisions of collection. As typical examples, music, literature, and film are categorized with genres based on their unique observable and objective characteristics [21, 25, 64]. Proper genre classification provides authoritative guidance to media audience in information retrieval, i.e., searching, browsing, locating and retrieving media items, based on common understanding on their characteristics [16]. It has become increasingly significant towards the distribution and marketing of cultural media, as well as to facilitate the learning and understanding of a particular domain [5, 32, 37]. However, mutual exclusivity and joint exhaustivity in genre classification, although being seen as the best practice for genre classification [9], are also considered impossible to define [35]. Thus, instead of strict and rigid taxonomies, flexible approaches in genre classification may be useful based on theories, such as the "family resemblance" notion by Wittgenstein [72], who use game as an example and *'see a complicated network of similarities overlapping and criss-crossing: sometimes overall similarities, sometimes similarities of detail'* to characterize such different similarities and relationships [50].

Regarding video game genre classification in the domain of game studies, previous studies tend to select various variables in order to reach stable definitions [66]. An early genre-like taxonomy given by Chris Crawford categorizes computer games into "skill-and-action" games and strategy games based on players' perceptual and cognitive skills [22]. Mark J.P. Wolf provides 42 categories of games based on gameplay and interactivity, excluding other elements, e.g., mood or theme [73]. Subsequently, with different focuses, many other studies also attempt to provide distinguishable categorization of video games in different ludic terms [1, 3, 20, 70]. On the other hand, other studies focus on the structural perspectives and provide meta-categories of video games [24, 33]. Summarized by Vargas-Iglesias, game studies in general investigate game genre from either an inductive or a deductive perspective, which, however, have not reached a consensual solution [66]. In addition, the strict and rigid taxonomies provided by the previous studies on video game genre classification are increasingly challenged

by the rapid evolution of video games together with the blending of multiple genre elements, where a more flexible approach could be useful [18]. Hence, concurring with Wittgenstein's statement and taking into account the overlapping similarity amongst contemporary video games, we characterize video games as a complex network connected by their similar attributes and distinguished by their uniqueness.

Together with the rapid development in data mining, many studies provide solutions towards genre classification, e.g., in music [59, 74] and movie [60, 77], with such computational approaches. Accordingly, the metadata of video games and players becomes increasingly important facilitating various studies towards understanding games and player behaviors [46, 58, 62]. However, limited studies have attempted such approaches on game genre classification. Faisal and Peltoniemi adopt Latent Dirichlet Allocation (LDA) topic modeling technique on game descriptions from online game databases and obtain 31 topics/genres [26]. The result provides a different game genre set but lacks further interpretation on the intertwined network-like connections between different genres. On the other hand, user generated tags, which allow end users to annotate and interact with information objects freely, provide a unique set of crowd-sourced metadata that facilitates the description and understanding of such information objects with decent agreement level to formal taxonomies [31, 67]. Specifically towards video games, Steam platform provides a unique user tagging feature enabling players to denote some aspects of the "aboutness" of the games [71]. Thereby, the connections among such tags (i.e., partial descriptions of games) forms the network of genres and sub-genres, instead of rigid taxonomies, requires further examination with approaches from network analysis.

Hence, in this study, we collect the user-generated tags metadata from Steam platform and construct a network of tags. Therein, each game tag is seen as a node of the network when an edge is formed between two tags provided both tags are annotated to the same game. Towards the purpose of understanding video game genre classification based on such game tag networks, we analyze the important game tags on the platform with network centrality measures as well as the grouping of the tags based on the network community detection. Such network analysis shall answer the following research questions: *RQ1. What are the most important game tags on Steam platform based on centrality measure?* and *RQ2. What are the major game tag communities via community detection?* The answers to the questions above shall lay the ground for an crowd-sourced understanding of video game genre classification.

The remainder of the article is organized as follows. Section 2 presents the related works in video game genre studies, the studies using Steam platform as data source, as well as the use of community detection methods tackling other classification-related issues. Section 3 briefly introduces network analysis methods adopted in this study. Section 4 describes the obtained Steam game tag dataset. Section 5 provides the analysis and reasoning of results. Section 6 further discusses relevant issues, including the limitation and future work of the study, concluding the article.

## 2 RELATED WORK

In the perspective of library and information science, the main purposes of genre classification are to identify the taxonomic identification of works, to enable collection and help users find similar items, and to promote the overall commercial marketing [18]. Although strict taxonomies of genre with mutual exclusivity and joint exhaustivity are ideal, such best practice is barely possible [9, 35]. For other forms of media, e.g., literature, films and music, a set of characteristics, such as, their styles, forms or patterns, are used to define their genres [21, 25]. For video games genre classification, the most common characteristic selected is their gameplay, which is the all-important quality factor and the pure interactivity of the games [36]. Focusing on the gameplay that the designer intend to advocate in particular games, many game genre classification are proposed that are either classified overly general (e.g., the "skill-and-action" and "strategy" genres by Chris Crawford [22]) or overly detailed (e.g., the 42 genres by Mark J.P. Wolf [73]). Thereafter, many other studies attempt to find the middle

ground but still focus on finding the mutual exclusivity and joint exhaustivity while also reason their proposal with focus on different perspectives. For example, Apperley focus on players' interaction and relations with the the game genres and propose four commonly used game labels [3]. Lee et al. use facet analysis and propose a 12-facet-and-358-foci scheme describing the game genre information [42]. Vargas-Iglesias focuses on the different functions within video games and presents four elemental genres and a way of configuring hybrid genres with binary function relations [66].

Many also argue that purpose shall be also taken into account when classifying genres. Johns suggests that genres shall be categorized the particular jobs they are used to accomplish instead of the structural components [34]. Other scholars on this matter also suggest that genres shall be defined as sets of communicative events sharing communicative purposes or communicative action with both purposes and elements of form [65]. For understanding game genres in such perspective, Bogost indicates that game genres are classified by the players through the on-screen effects and controllable dynamics they experience [11]. On the other hand, other scholars also suggest to understand game genre as a socially constructed phenomena instead of a clear-cut taxonomy. Clearwater indicates that formal and aesthetic considerations, industrial and discursive context, and social meaning and cultural practice shall all be taken into account regarding video game genres [20]. Arsenault also emphasizes that the genre of a game is tied not to a checklist of features, but to the phenomenological, pragmatic deployment of actions through the gameplay experience, as gameplay is both functional and aesthetic [4].

Despite the studies in video game genre classification that provides various classification outcomes, the issues of such classification being rigid and falling short at contributing to the original purpose of classification persist. Considering the primary purpose of genre classification being to help users find similar items, the previous classifications perform ineffectively when games with different characteristics located in same genre lacking concrete identification criteria [18]. Such lack of concrete definition then results in the heavily overloaded genre labels that representing multi-dimensional information [42]. Towards such end, metadata are used to describe video games and interactive media. Lee et al. introduce a schema of 16 elements to formally describe video games based on a user-centered design approach [44]. Despite the importance of the works towards preserving and retrieving video game legacy as well as the usefulness of metadata reflecting the characteristics of video games [41], it falls short on providing approaches towards game genre classification based on the analysis of such metadata.

Steam is one of the most popular digital game distribution platforms, offering various services including video game purchasing and downloading, digital rights management, online game matchmaking, video streaming, forum and social networking, etc. It has also drawn attention from the academia targeting the studies on multiple aspects of the Steam community, video games and the players due to the detailed metadata it provides. Regarding player behaviors, Sifa et al. analyze the players' different playtime frequency distribution and investigate their engagement and cross-game behavior on Steam [57, 58]. Lim and Harrell examine the players' social identity, as well as the connection between players' behaviors on maintaining their profiles and their social network [46]. Regarding video games, Slivar et al. analyze the the impact of game types and video adaptation strategies on the quality of experience via a case study on Steam platform [62]. Lin et al. focus on the video game development and maintenance practices and analyze the urgent update strategy of popular games on Steam [47]. Windleharth et al., conduct a conceptual analysis on the user-generated game tags on Steam and propose a categorization of them according to the Video Game Metadata Schema (VGMS) category [43, 71]. Furthermore, other perspectives regarding player communities [8], game reviews [49], game recommendation mechanisms [75], and etc. have also been studied using the data obtained from Steam.

On the other hand, network analysis and community detection methods have been used in various domain tackling the classification related issues. For example, Siew uses the Louvain community detection method to extract communities in the phonological network towards understanding the dynamics of activation spread among words and the mechanisms that underlie language acquisition and the evolution of language [56]. Sokolova

et al. use similar method to classify Android applications based on the analysis of the permission requests network that provides information about the application's behavior [63]. Such methods are also used in biology and music studies in terms of classification [27, 78].

## 3 METHODOLOGY

In this study, two classic centrality measures: closeness centrality and betweenness centrality [28], are applied to the obtained game tag network. In addition, PageRank, a popular algorithm measuring the importance of website pages [15], is also adopted herein to measure the importance of game tag vertices, compared with the results of the centrality measures. On the other hand, Louvain method for community detection, a method to extract communities from large networks [10], is used to obtain the communities of game tags.

### 3.1 Closeness and Betweenness Centrality

Closeness centrality is an importance index based on the geodesic distances from the network vertices to all others. It is a metric used to identify how long it shall take for information to travel from a particular vertex to the others in the network, i.e., how close is it to them [53]. Let $V$ be the set of vertices of a given network, where $i$ is a particular vertex within. The geodesic distance between $i$ and another vertex $j \in V$ is denoted as $d_G(i, j)$. Thus, as the closeness centrality is defined as the inverse of the average distance [7], the closeness centrality of $i$, $C_C(i)$, is calculated as follows.

$$C_C(i) = \frac{1}{\sum_{j \in V} D_G(i, j)} \tag{1}$$

Betweenness centrality, indicating the "brokering positions between others that provides opportunity to intercept or influence their communication" [14], is based on the shortest paths through a particular vertex. Herein, the geodesic path of two individual vertices in the network is defined as the shortest path between them. For a particular vertex $i \in V$, the number of geodesic paths between another two vertices $h, j \in V$ via vertex $i$ is denoted as $g_{hij}$. Meanwhile, the number of geodesic paths from $h$ to $j$ is denoted as $g_{hj}$. Then, the betweenness centrality of $i$, $C_B(i)$, is calculated as follows.

$$C_B(i) = \sum_{j,h \neq i} \frac{g_{hij}}{g_{hj}} \tag{2}$$

Both Betweenness Centrality and Closeness Centrality calculate the shortest paths of a vertex to the rest of the vertex pairs in relative large network [12]. Betweenness centrality measures the others' dependence on a specific vertex[13], to define which vertex has the most control. Closeness centrality measure the access efficiency of the specific vertex to the other vertices [76], to define which vertex can most easily reach the rests of vertices between vertices or sub-vertices. Regarding the network of game tags, the tags with the high closeness centrality are the ones more close to the other tags in the network. Hence, such tags are more common applied to games together with other tags. On the other hand, the game tags of high betweenness centrality are the ones critically linking two sets of tags that are tend to be separate from each other. Thus, when removing such tags, we tend to have separate sets of tags across which seldom be applied to same games.

### 3.2 PageRank

PageRank is the core designing concept of Google search engine. It assigns universal ranks to web pages based on a weight-propagation algorithm [15]. The web page ranking mechanism can be described as: a web page is assigned high rank if the sum of the ranks of its backlinks is high, which covers both the case when a page has many backlinks and when a page has a few highly ranked backlinks. We assume web page $w$ has $k$ pages (i.e.,

$w_1$, $w_2$, ..., $w_k$) linking to it when there are totally $N$ web pages. We also define a damping factor $d \in (0, 1)$ as the probability at each page the "random surfer" will get bored and request another random page (normally set at 0.85) [15]. Let $C(w)$ be the number of links going out of $w$ while the the PageRank value of $w$, $PR(w)$ is then calculated as follows.

$$PR(w) = \frac{1-d}{N} + d \sum_{i=1}^{k} \frac{PR(w_i)}{C(w_i)} \tag{3}$$

Specifically regarding the Steam game tags, one is assigned high rank when it co-occurs with many other tags in games or it it co-occurs with highly ranked tags.

### 3.3 Network Modularity and Community Detection

It is common that very diverse systems in various domains can be described as complex network, when community structure is a topological property of networks [2, 55]. A network community (also referred to as module or cluster) is a group of vertices that have denser connections with the members of the group than the connections with the remainder of the network [30, 55]. Hence, community detection is to identify such communities of a network in order to reveal the valuable information of its structure and functionalities. On the other hand, modularity is a popular measure for the structure of networks, as it measures the strength of division of a network into communities [52]. High-modularity networks have dense connections between the vertices within communities and comparatively sparse connections between vertices between different communities. Herein, we adopt the Louvain method that extracts the community structure of a particular large weighted network when its modularity value is optimized [10].

A network is defined by $V$ and $E$ being the set of vertices and edges of the network. We assume $m$ be the number of communities the network is partitioned into. Let $l_k$ be the number of edges between any two vertices from the $k$-th community. Meanwhile, let $d_k$ be the sum of degree of all those vertices. The network modularity $Q$ is then calculated as follows.

$$Q = \sum_{k=1}^{m} [\frac{l_k}{|E|} - (\frac{d_k}{2|E|})^2] \tag{4}$$

According to the Louvain community detection method, we firstly assign each vertex to a community when $Q$ is maximized. When moving vertex $i$ to community $C$, the gained $\Delta Q$ is then calculated as follows.

$$\Delta Q = \frac{\sum_C + k_i^C}{2n} - (\frac{\sum_{\widehat{C}+k_i}}{2n})^2 - [\frac{\sum_C}{2n} - (\frac{\sum_{\widehat{C}}}{2n})^2 - \frac{k_i}{2n}] \tag{5}$$

In the above equation, $k_i$ is the sum of weighted edges incident to $i$; $k_i^C$ is such sum of the edges from $i$ to vertices in community $C$; $\sum_C$ is sum of the weighted edges in $C$; $\sum_{\widehat{C}}$ is such sum of the edges incident to vertices in $C$. Let $n$ be the sum of the weights of all the edges of the network. Continuously, the method changes the structure of the communities by moving vertices from one community to another and calculating $\Delta Q$ until $\Delta Q$ is significantly improved [23].

## 4 DATA

We choose Steam platform as the source of data crawling as it contains the largest video game collection compared with other game platforms on PC, e.g., Origin, GOG, Uplay, and so on. On the other hand, console-exclusive games are of limited amount compared to the volume of games on Steam, when the majority of games on consoles are also published on Steam. On the other hand, despite the mobility of mobile devices and unique ways of interaction for mobile players [45], mobile platform contains limited unique game genres that unique to the

devices [51]. Thus, the data obtained from Steam platform shall be seen statistically representative for PC and console based video games. In this study, we crawl all 378 game tags from Steam platform “Popular Tags" page[1] using BeautifulSoup[2]. For any two different game tags, we identify they are undirectedly connected when at least one game on Steam contains both of the two tags. In addition, each connection between two particular game tags are weighted by the number of games that contain both tags. For example, as 1119 games on Steam have both the tag “Adventure" and “Open World", the undirected connection between these two tags is weighted 1119.


Fig. 1. The Top Connected Tags for "FPS"

Furthermore, we filter the weighted connections between the obtained game tags by selecting only the top weighted connections based on the “Narrow by Tags" recommendation function of Steam platform. For example, Figure 1 shows the recommended most commonly connected tags for the "FPS" tag, where the number of games containing both tags are also shown. Thereby, we only select the top weighted connections shown in the “Narrow by Tag" for each tag with duplicates filtered. As 38 tags out of the 378 contain no recommended tags, we drop them considering these tags hardly connect any other tags in any games. Another 14 tags for utilitarian software products, e.g., “Utilities", “Design & Illustration", ‘Animation & Modeling" and so on, are also eliminated due to their irrelevance to games. For the remaining 326 tags, we obtain 3035 weighted tag connections.

Using Gephi [6], a popular open-source network analysis and visualization software, we visualize the obtained game tag connections into a graph with 326 vertices and 3035 edges. The average degree of the graph is 18.62 when the average weighted degree is 4703.656. Among the tags, the “Indie" tag has the highest degree of 300 and the highest weighted degree of 171430, while seven tags have the lowest degree of 1 and tag “Gambling" has the lowest weighted degree of 21. The diameter of the network is 4 with an average path length of 2.005. The graph density is 0.057.

## 5 RESULTS

Based on the game tag network constructed, we calculate the weighted degree, closeness centrality, betweenness centrality and PageRank for each tag. By doing so, the important game tags can be identified by the above metrics. On the other hand, via the Louvain community detection method, we also identify the communities of the network, based on which the core game genres can be summarized by the game tags contained within.

[1]https://store.steampowered.com/tag/browse/
[2]https://www.crummy.com/software/BeautifulSoup/

## 5.1 The Important Game Tags on Steam

The obtained results show that five game tags, i.e., “Indie", “Action", “Adventure", “Singleplayer" and “Casual", rank the highest in all above mentioned metrics. Shown in Figure 2, which shows the Top 15 tags by every metric, these five game tags have significantly higher value than the rest ten regardless.

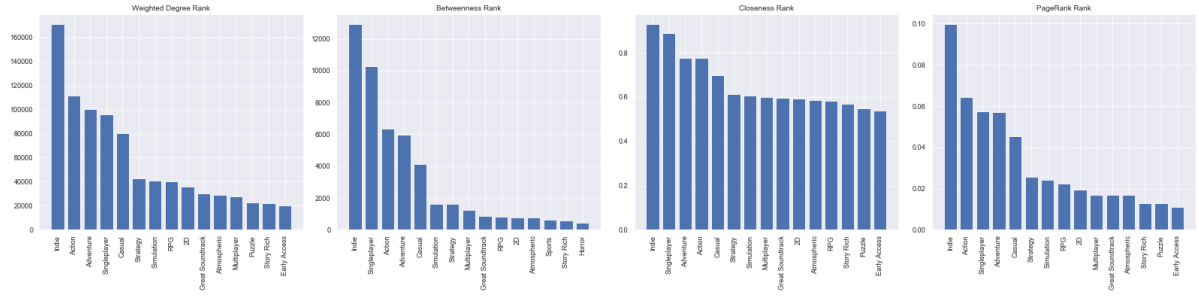


Fig. 2. Importance Rank of Game Tags based on Weighted Degree, Betweenness, Closeness and PageRank

The results indicate that each of these tags associates with the majority number of other tags with the number of games including them being the largest. Furthermore, each of these tags occur most frequently together with the other four or the less important ones. Additionally, the number of games with at least one of these tags rank also the highest. We can thus suggest that these five tags are the most commonly adopted ones on Steam platform. Taking into account the category of Steam tags with the Video Game Metadata Schema (VGMS) category [43, 71], “Action" and “Adventure" are the core gameplay related tags, while “Indie" (Production) and “Singleplayer" (Number of Players) are concerning the other perspectives. “Casual" is a complex notion regarding games, the players and their playing styles and manners [38], which is also regarding a different perspective from gameplay.

Indie games, often referring to “Independent games", has become a increasingly popular concept in the game industry and game academia. Commonly, the “Indie" concept is used to distinguish the independently developed games from the “AAA" or “mainstream" ones; however, such implicit categorization evokes further arguments [61]. Garda and Grabarczyk argue that indie games shall be seen “independent" from three perspectives: financial independence, creative independence and publishing independence, by belonging to at least one of which a game shall be classified as independent [29]. In addition, the authors also argue that the concept of indie game refers to a broad understanding of independent games including the adoption of digital distribution, the preference of retro style, the small budge and team, etc. Steam, as a online distribution channel for video games, enables independent developers to publish games easily. Furthermore, the “Early Access" mechanism of Steam also support independent developers financially by enable them continuously maintaining their “incomplete" games by earning early commitments from their players [48]. Therefore, the thriving number of games with “Indie" tag is, to some extent, reasonable.

The action and adventure games have long been an important genre, which is included in many game genre classifications. In Crawford’s early classification of “Skill & Action" games and “Strategy" games, adventure games are seen as part of the strategy games, as the player requires proper strategies to move within complex world, find and manage tools, overcome obstacles and reach goals [22]. Another reason to such classification is that the adventure games at that time were majorly text-based (e.g., *Zork*). In Wolf’s 42 gameplay-based genres, adventure game is categorized similarly with emphasis on obstacle variety and free exploration, while action game is not specifically categorized [73]. Thereafter, the other studies on game genre classification tend to categorize action

game as a relatively broader genre when adventure game being assimilated towards role-playing games (RPG) [3]. As the game industry grows and the information technology advances, the boundary of action and adventure game grows ambiguous when players mostly play roles in an exploring adventure and meanwhile combat through obstacles by actions [19]. “Action" and “Adventure" become the easiest labels to attach to games, as, technically, we can see all games as action games due to the fact that all games involve physical input interactions that result in actions on the screen, meanwhile we can also see all games as adventure because all players take on the roles of imaginary characters who engage in adventures [18]. Therefore, these given factors, to a certain extent, rationalize the surprising result of “Action" and “Adventure" being seen more as universal tags than genre-specific ones. It should be hereby emphasized that we mean not to assert that “Action-Adventure" shall be seen as a common game element, but instead only provide findings to reflect the players’ existing tendency of seeing both tags applicable to multiple game types.

Casual games, described as the ones with less complicated controls and complexity in gameplay, have been exceeding in game market since a decade ago [68]. It is hard to fit casual games into a particular genre based on gameplay (like action and adventure), when the notion “casual" herein can be understood from multiple perspectives in terms of specified terminologies, including casual games (i.e., games with appealing content, simple controls, easy-to-learn gameplay, etc.), casual gaming (i.e., a casual gaming attitude), casual playing (i.e., casual play session), casual gamer (i.e., a player playing a not-necessarily-casual game in casual manner), and casual game player (i.e., a player playing a casual-labelled game) [38, 39]. Therefore, it is common that a number of games from various gameplay-based genres are labeled “casual" regarding any of the above mentioned ways of thinking.

## 5.2 Genres from Community Detection

It has been acknowledged that nodes that are connected to many other vertices across a network can disturb the community structures [17, 69]. Thus, in order to better detect the communities from the game tags network, we remove the above mentioned five core tags from the network due to their high degree. Meanwhile, another 15 vertices are detected only connecting to these five tags, including “Pinball", “Gambling", “Typing", etc. which are removed as well. Thereafter, we apply the Louvain method on the remaining network with 306 vertices and 1813 edges. With the feature of randomizing the initial moving node for the Louvain method in Gephi, we obtain the highest modularity score of $Q = 0.414$ with four communities (shown in Figure 3) detected via over 100 rounds of testing. The modularity result is largely improved from the result ($Q = 0.150$) of a previous experiment with the five high-degree vertices included, which indicates a comparatively strong community structure.

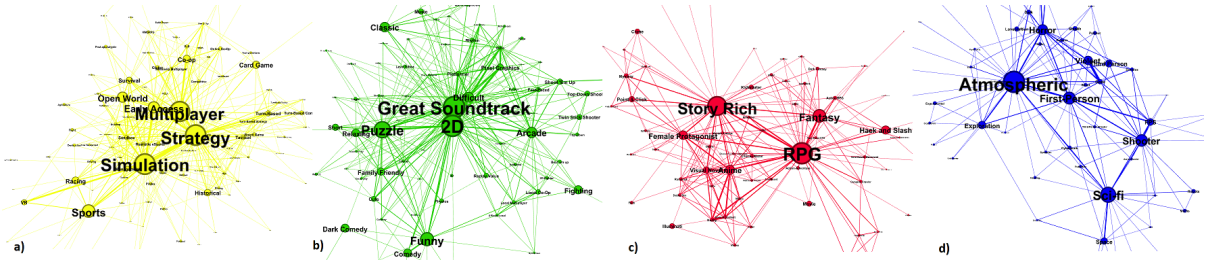


Fig. 3. The Detected Communities of Tags

Based on the games tags contained in each detected community, we summarize the four major game genres based on the high-centrality gameplay-defining tags within. Via the Video Game Metadata Schema (VGMS)

category [43] and the Steam tag categorization based on information presentation [71] as references, we describe each genre in details as follows.

*5.2.1 Genre 1: Strategy & Simulation Games.* This game genre contains 104 game tags (34.0%) and 468 connections. With the closeness centrality, betweenness centrality and PageRank, the Top 10 tags of this genre are listed as follows.

Table 1. Genre 1 Top Tags

| | **Closeness** | **Betweenness** | **PageRank** |
|---|---|---|---|
| 1 | Strategy | Simulation | Strategy |
| 2 | Simulation | Strategy | Simulation |
| 3 | Multiplayer | Multiplayer | Multiplayer |
| 4 | Open World | Sports | Early Access |
| 5 | Early Access | Early Access | Open World |
| 6 | Survival | Open World | Co-op |
| 7 | Political | Card Game | Turn-based |
| 8 | Medieval | Co-op | Sand-box |
| 9 | Historical | Racing | Sports |
| 10 | Realistic | Historical | Survival |

We summarize this genre as the *Strategy & Simulation* due to the fact that these two gameplay-defining tags rank highest in all three importance metrics. The other gameplay tags in this genre, e.g., “Sports", “Racing", “Survival", etc., can be understood as a more specific type of simulation-oriented game with, certainly, strategy needed to win. On the other hand, this genre also contains other types of tags, such as, “Open World" (Progression), “Multiplayer" (Number of Players), “Realistic" (Mood), “Turn-based" (Pacing) etc. indicating the other perspectives of the games in this genre.

*5.2.2 Genre 2: Puzzle & Arcade Games.* This genre contains 79 game tags (25.8%) and 342 connections. This genre can be summarized as *Puzzle & Arcade* due to the important gameplay tags.

Table 2. Genre 2 Top Tags

| | **Closeness** | **Betweenness** | **PageRank** |
|---|---|---|---|
| 1 | 2D | Great Soundtrack | 2D |
| 2 | Great Soundtrack | 2D | Great Soundtrack |
| 3 | Puzzle | Puzzle | Puzzle |
| 4 | Funny | Funny | Difficult |
| 5 | Comedy | Arcade | Funny |
| 6 | Relaxing | Difficult | Pixel Graphics |
| 7 | Classic | Classic | Arcade |
| 8 | Masterpiece | Fighting | Platformer |
| 9 | Family Friendly | Dark Comedy | Retro |
| 10 | Reply Value | Comedy | Family Friendly |

The convergence of “Puzzle" and “Arcade" games into one genre is due to the large weight between each of the two tags and the “2D" and “Great Soundtrack" tags of high centrality. The other less important gameplay tags

include “Classic", “Fighting", and so on, which can be seen as either a similar overlapping concept or a sub-genre. Furthermore, “Difficult" (Evaluation), “Funny" and “Relaxing" (Mood), “Retro" (Visual Style), “Family Friendly" (Rating) are mostly connected to this genre. Importantly, “2D" and “Great Soundtrack" are the two tags with high importance values, indicating the most intuitive characteristics of the games in this genre.

*5.2.3 Genre 3: RPG Games.* This genre contains 68 game tags (22.2%) and 232 connections. Due to the most important tag in this genre being “RPG", we summarize it as the *RPG* genre.

Table 3. Genre 3 Top Tags

| | **Closeness** | **Betweenness** | **PageRank** |
|---|---|---|---|
| 1 | RPG | RPG | RPG |
| 2 | Story Rich | Story Rich | Story Rich |
| 3 | Fantasy | Fantasy | Fantasy |
| 4 | Choices Matter | Anime | Anime |
| 5 | Text-based | Female Protagonist | Female Protagonist |
| 6 | Dark Fantasy | Hack and Slash | Visual Novel |
| 7 | Kickstarter | Visual Novel | Nudity |
| 8 | Action RPG | Point & Click | Sexual Content |
| 9 | Mythology | Movie | Point & Click |
| 10 | Crime | Illuminati | Mystery |

The main gameplay of this genre include role playing and story, while the other tags with high importance values are mostly regarding the particular story themes (e.g., “Fantasy", “Crime"), main characters (e.g., “Female Protagonist"), game mechanics (e.g., “Hack and Slash", “Point & Click"), or rating (e.g., “Nudity", “Sexual Content"). “Visual Story" is the other gameplay related tag, which can be seen as a story-focus RPG without combats.

*5.2.4 Genre 4: Shooter Games.* This genre contains 55 game tags (18.0%) and 155 connections. Similar to the previous genre, we summarize it as *Shooter* genre by the highest valued gameplay related tag.

Table 4. Genre 4 Top Tags

| | **Closeness** | **Betweenness** | **PageRank** |
|---|---|---|---|
| 1 | Atmospheric | Atmospheric | Atmospheric |
| 2 | Sci-fi | Sci-fi | Shooter |
| 3 | Exploration | First-Person | Sci-fi |
| 4 | First-Person | Shooter | First-Person |
| 5 | Space | Horror | Horror |
| 6 | Dystopian | Violent | Violent |
| 7 | Third Person | Exploration | FPS |
| 8 | Hacking | Third Person | Gore |
| 9 | Underwater | FPS | Exploration |
| 10 | Futuristic | Space | Space |

Based on the top ranking tags in this genre, “First-Person" and “Third Person", are the main sub-genres of shooter games. The major themes of shooter games include “Sci-fi", “Space", “Dystopian", and so on, which evokes the players’ mood, including “Horror", “Atmospheric", “futuristic", etc.

The above game tag community detection results can be seen aligning closely with Vargas-Iglesias' study on game genre classification focusing on the understanding of gameplay category as a functional construction, but with slight differences [66]. The author proposes four elemental game genres (i.e., Action, Strategy, Puzzle and RPG) based on, respectively, four game functions (i.e., Intuitive, Formal, Inductive, and Deductive). Furthermore, the study also indicates that we can distinguish a particular game by the elemental genres according to the one single function configuring the game, while games of hybrid genres are configured by a binary functional relations, i.e., either with two separated factors or with one factor submitted to the other.

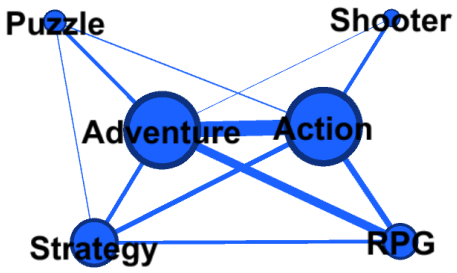


Fig. 4. The Connections between Core Tags

The key difference between Vargas-Iglesias' elemental genres and our community detection result is the universalized "Action" and "Adventure" tag with the reasons of such phenomena presented previously. By observing the connections between the six tags (shown in Figure 4), we find all the four genre-defining tags are linked to both "Adventure" and "Action" tags, suggesting that each combination covers a considerable number of games. Compared to a broader notion of "Action", the "shooter" games can be perceived as more specifically motor-skill-oriented and distinguishable from the other genres. Thus, such result can be seen as a validation of Vargas-Iglesias' function-oriented genre classification by confirming that games are likely to be perceived and distinguished towards such genres according to players' collective intelligence reflected by the generated tags on Steam platform.

However, despite the agreement of such results to previous genre classification, due to the nature of the applied network analysis method, the results shall only reflect the existence of a phenomenon that video games share the tendency of belonging to one of the four genres and most likely contain other elements as well. The obtained game tag network shows that all the detected tag communities are inter-connected via various tags indicating the common existence of cross-genre games. Thus, this study is to show that, facilitated by such network analysis, we can detect such connections between the tags that reflect elemental genres and the other relevant information, such as, themes, narratives, visual styles, players' mood, pacing and so on [71]. For example, "Historical" and "Medieval" themes are more often adopted for strategy and simulation games, when "Sci-fi" and "Futuristic" themes are for shooter games; "2D" visual style and "Great Soundtrack"s are often adopted by puzzle and arcade games towards "Relaxing" and "Funny" experiences, when RPG games often prefer "Fantasy" theme and "Choice Matter" mechanics. More importantly, such perception of game genres aligns with and emphasizes the nature of the game genre classification being a *complicated network of overlapping similarities* [72].

## 6 DISCUSSION AND CONCLUSION

This study provides a new approach of video game genre classification via the network analysis of the user-generated game tags on Steam with centrality analysis and community detection. It contributes to game genre studies by offering a new way of understanding the complexity and intertwined connections of game genres

with a set of interesting findings. Firstly, we find that players tend to apply the tag “Action" and “Adventure" to various games across genres, which results in the phenomenon of having these two tags commonly connected to majority of other tags. The reason for the phenomena is, agreeing to the statement from Clarke et al. [18] that any physical input interactions resulting in actions on the screen can be seen as “Action" while any story in game with player taking the role of an imaginary character can be seen as “Adventure". Meanwhile, “Indie", “Singleplayer" and “Casual" are the other three commonly applied tags that are connected to majority of other genre-reflecting tags. The reason for such phenomenon is that these tags reflect the "non-gameplay" aspects of games. Secondly, we find video games can be roughly classified into four genres, i.e., *Strategy & Simulation*, *Puzzle & Arcade*, *RPG* and *Shooter*, which aligns with the results of Vargas-Iglesias' four elemental genres [66]. The main difference is that, due to the notion of “Action" being often perceived universalized in terms of video games, we find “shooter" being the important tag for the genre reflecting the motor-skill-based mechanics. Moreover, puzzle games and arcade games are found closely connected by the common application of 2D visual style and great soundtracks as media enhancement.

Importantly, we argue that video games, in terms of genres, form a complicated network where they share a particular set of similarities and also differ in various ways. Thus, despite the usefulness of genre classification in general, such perception of “network-ish" game genres shall have its position in the game studies domain. The obtained network of game tags intuitively shows that it is nearly impossible to find any game that can be classified into a single genre, when it most likely contains particular elements that similar to ones from other genres. What needs to be emphasized is that the preliminary findings of this study do not necessarily reject the previous genre classification or propose the the obtained classification results as a new better set of genres. Instead, this study aims to propose an alternative approach of understanding game genres via network analysis which better reflects the such complex connections underneath. It shall lay the ground for and encourage the further investigation of such intertwined connections between the game genres (reflected by game tags) of unevenly defined abstraction levels and different focuses with the repertoire of network analysis and other data-driven approaches.

Despite the preliminary findings, this study can still be improved towards the following ends. This study only contains data from Steam platform and does not include data from other platforms, such as, mobile devices, consoles, and other PC-based platforms. Though we argue the Steam platform data being statistically representative, it can still be seen as a major threat to validity. For example, location-based games, e.g., *Pokemon Go*[3][54], can be seen as a unique game genre but is not covered herein. On the other hand, the understanding of user-generated tags as well as the players' perception regarding game genres shall be further investigated via future empirical studies. A potential comparison between the perception of game genres from players and game analysts shall also contribute to such end. Furthermore, a more quantified dataset regarding the prioritized game tags for each particular game can lead to more thorough results with fruitful details. In addition, other data sources, such as the players' reviews and comments on game forums, can also contribute to the such understanding. Regarding the methodologies, Louvain community detection method falls short in detecting small communities of networks, where the other community detection methods can be applied for meaningful comparison [40]. By doing so, we could expect a hierarchical structure of game genre classification model working more effectively towards game recommendation and player community recommendation. Furthermore, our future work shall also include using other machine learning methods to categorize video games towards potential genre classification, provided such data is available.

## REFERENCES

[1] Espen Aarseth, Solveig Marie Smedstad, and Lise Sunnanå. 2003. A multidimensional typology of games.. In *DiGRA Conference*.

[2] Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Reviews of modern physics* 74, 1 (2002), 47.

[3] https://www.pokemongo.com/

[3] Thomas H Apperley. 2006. Genre and game studies: Toward a critical approach to video game genres. *Simulation & Gaming* 37, 1 (2006), 6–23.

[4] Dominic Arsenault. 2009. Video game genre, evolution and innovation. *Eludamos. Journal for Computer Game Culture* 3, 2 (2009), 149–176.

[5] Bruce A Austin and Thomas F Gordon. 1987. Movie genres: Toward a conceptualized model and standardized definitions. *Current Research in Film: Audiences, Economics and the Law* 4 (1987), 12–33.

[6] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. 2009. Gephi: an open source software for exploring and manipulating networks. In *Third international AAAI conference on weblogs and social media*.

[7] Alex Bavelas. 1950. Communication patterns in task-oriented groups. *The Journal of the Acoustical Society of America* 22, 6 (1950), 725–730.

[8] Roi Becker, Yifat Chernihov, Yuval Shavitt, and Noa Zilberman. 2012. An analysis of the steam community network evolution. In *Electrical & Electronics Engineers in Israel (IEEEI), 2012 IEEE 27th Convention of*. IEEE, 1–5.

[9] Clare Beghtol. 2000. The concept of genre and its characteristics. *Bulletin of the American Society for Information Science and Technology* 27, 2 (2000), 17–17.

[10] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (2008), P10008.

[11] Ian Bogost. 2007. *Persuasive games*. Vol. 5. Cambridge, MA: MIT Press.

[12] Ulrik Brandes. 2001. A Faster Algorithm for Betweenness Centrality. *Journal of Mathematical Sociology* 25 (2001), 163–177.

[13] Ulrik Brandes, Stephen Borgatti, and Linton Freeman. 2016. Maintaining the duality of closeness and betweenness centrality. *Social Networks* 44 (01 2016), 153–159. https://doi.org/10.1016/j.socnet.2015.08.003

[14] Ulrik Brandes, Stephen P Borgatti, and Linton C Freeman. 2016. Maintaining the duality of closeness and betweenness centrality. *Social Networks* 44 (2016), 153–159.

[15] Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems* 30, 1-7 (1998), 107–117.

[16] Lois Mai Chan and Athena Salaba. 2015. *Cataloging and classification: an introduction*. Rowman & Littlefield.

[17] Pin-Yu Chen and Alfred O Hero. 2015. Deep community detection. *IEEE Transactions on Signal Processing* 63, 21 (2015), 5706–5719.

[18] Rachel Ivy Clarke, Jin Ha Lee, and Neils Clark. 2017. Why video game genres fail: A classificatory analysis. *Games and Culture* 12, 5 (2017), 445–465.

[19] Rachel I Clarke, Jin Ha Lee, and Stephanie Rossi. 2015. A Qualitative investigation of users' video game information needs and behaviors. *School of Information Studies - Faculty Scholarship* 166 (2015). https://surface.syr.edu/istpub/166

[20] David Clearwater. 2011. What defines video game genre? Thinking about genre study after the great divide. *Loading...* 5, 8 (2011).

[21] Pam Cook. 2007. *The cinema book*. British Film Institute.

[22] Chris Crawford. 1984. The art of computer game design. (1984).

[23] Pasquale De Meo, Emilio Ferrara, Giacomo Fiumara, and Alessandro Provetti. 2011. Generalized louvain method for community detection in large networks. In *2011 11th International Conference on Intelligent Systems Design and Applications*. IEEE, 88–93.

[24] Christian Elverdam and Espen Aarseth. 2007. Game classification and game design: Construction through critical analysis. *Games and Culture* 2, 1 (2007), 3–22.

[25] Franco Fabbri. 1982. A theory of musical genres. Two applications. (1982).

[26] Ali Faisal and Mirva Peltoniemi. 2018. Establishing video game genres using data-driven modeling and product databases. *Games and Culture* 13, 1 (2018), 20–43.

[27] Stefano Ferretti. 2018. On the complex network structure of musical pieces: analysis of some use cases from different music genres. *Multimedia Tools and Applications* 77, 13 (2018), 16003–16029.

[28] Linton C Freeman. 1978. Centrality in social networks conceptual clarification. *Social networks* 1, 3 (1978), 215–239.

[29] Maria B Garda and Paweł Grabarczyk. 2016. Is every indie game independent? Towards the concept of independent game. *Game Studies* 16, 1 (2016).

[30] Michelle Girvan and Mark EJ Newman. 2002. Community structure in social and biological networks. *Proceedings of the national academy of sciences* 99, 12 (2002), 7821–7826.

[31] Scott A Golder and Bernardo A Huberman. 2006. Usage patterns of collaborative tagging systems. *Journal of information science* 32, 2 (2006), 198–208.

[32] Albert N Greco. 2013. *The book publishing industry*. Routledge.

[33] Aki Järvinen. 2008. *Games without frontiers: Theories and methods for game studies and design*. Tampere University Press.

[34] Ann M Johns. 1997. *Text, role and context: Developing academic literacies*. Cambridge University Press.

[35] Kevin P Jones. 1973. The environment of classification: the concept of mutual exclusivity. *Journal of the American Society for Information Science* 24, 2 (1973), 157–163.

[36] Jesper Juul. 2011. *Half-real: Video games between real rules and fictional worlds*. MIT press.

[37] Heather Kay and Tony Dudley-Evans. 1998. Genre: What teachers think. (1998).

[38] Jussi Kuittinen, Annakaisa Kultima, Johannes Niemelä, and Janne Paavilainen. 2007. Casual games discussion. In *Proceedings of the 2007 conference on Future Play*. ACM, 105–112.

[39] Annakaisa Kultima. 2009. Casual game design values. In *Proceedings of the 13th international MindTrek conference: Everyday life in the ubiquitous era*. ACM, 58–65.

[40] Andrea Lancichinetti and Santo Fortunato. 2009. Community detection algorithms: a comparative analysis. *Physical review E* 80, 5 (2009), 056117.

[41] Jin Ha Lee, Rachel Ivy Clarke, and Andrew Perti. 2015. Empirical evaluation of metadata for video games and interactive media. *Journal of the Association for Information Science and Technology* 66, 12 (2015), 2609–2625.

[42] Jin Ha Lee, Natascha Karlova, Rachel Ivy Clarke, Katherine Thornton, and Andrew Perti. 2014. Facet analysis of video game genres. *iConference 2014 Proceedings* (2014).

[43] Jin Ha Lee, Andrew Perti, Rachel Ivy Clarke, Travis W Windleharth, and Marc Schmalz. 2017. *UW/SIMM Video Game Metadata Schema Version 4.0*. http://gamer.ischool.uw.edu/official_release/

[44] Jin Ha Lee, Joseph T Tennis, Rachel Ivy Clarke, and Michael Carpenter. 2013. Developing a video game metadata schema for the Seattle Interactive Media Museum. *International journal on digital libraries* 13, 2 (2013), 105–117.

[45] Xiaozhou Li and Zheying Zhang. 2015. A User-App Interaction Reference Model for Mobility Requirements Analysis. In *ICSEA 2015, The Tenth International Conference on Software Engineering Advances*. 170–177. https://academic.microsoft.com/paper/2604476080

[46] Chong-U Lim and D Fox Harrell. 2014. Developing Social Identity Models of Players from Game Telemetry Data.. In *AIIDE*.

[47] Dayi Lin, Cor-Paul Bezemer, and Ahmed E Hassan. 2017. Studying the urgent updates of popular games on the steam platform. *Empirical Software Engineering* 22, 4 (2017), 2095–2126.

[48] Dayi Lin, Cor-Paul Bezemer, and Ahmed E Hassan. 2018. An empirical study of early access games on the Steam platform. *Empirical Software Engineering* 23, 2 (2018), 771–799.

[49] Dayi Lin, Cor-Paul Bezemer, Ying Zou, and Ahmed E Hassan. 2018. An empirical study of game reviews on the Steam platform. *Empirical Software Engineering* (2018), 1–38.

[50] Anthony Manser. 1967. Games and Family Resemblances. *Philosophy* 42, 161 (1967), 210–225. https://doi.org/10.1017/S0031819100001297

[51] Frans Mäyrä. 2015. Mobile games. *The International Encyclopedia of Digital Communication and Society* (2015), 1–6.

[52] Mark EJ Newman. 2006. Modularity and community structure in networks. *Proceedings of the national academy of sciences* 103, 23 (2006), 8577–8582.

[53] Kazuya Okamoto, Wei Chen, and Xiang-Yang Li. 2008. Ranking of Closeness Centrality for Large-Scale Social Networks. In *Frontiers in Algorithmics*, Franco P. Preparata, Xiaodong Wu, and Jianping Yin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 186–195.

[54] Janne Paavilainen, Hannu Korhonen, Kati Alha, Jaakko Stenros, Elina Koskinen, and Frans Mayra. 2017. The Pokémon GO experience: A location-based augmented reality mobile game goes mainstream. In *Proceedings of the 2017 CHI conference on human factors in computing systems*. ACM, 2493–2498.

[55] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. 2004. Defining and identifying communities in networks. *Proceedings of the national academy of sciences* 101, 9 (2004), 2658–2663.

[56] Cynthia SQ Siew. 2013. Community structure in the phonological network. *Frontiers in psychology* 4 (2013), 553.

[57] Rafet Sifa, Christian Bauckhage, and Anders Drachen. 2014. The Playtime Principle: Large-scale cross-games interest modeling.. In *CIG*. 1–8.

[58] Rafet Sifa, Anders Drachen, and Christian Bauckhage. 2015. Large-scale cross-game player behavior analysis on steam. *Borderlands* 2 (2015), 46–378.

[59] Carlos N Silla, Alessandro L Koerich, and Celso AA Kaestner. 2008. A machine learning approach to automatic music genre classification. *Journal of the Brazilian Computer Society* 14, 3 (2008), 7–18.

[60] Gabriel S Simões, Jônatas Wehrmann, Rodrigo C Barros, and Duncan D Ruiz. 2016. Movie genre classification with convolutional neural networks. In *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 259–266.

[61] Bart Simon. [n.d.]. Indie Eh? Some Kind of Game Studies. *Loading...* 7, 11 ([n. d.]).

[62] Ivan Slivar, Mirko Suznjevic, and Lea Skorin-Kapov. 2015. The impact of video encoding parameters and game type on QoE for cloud gaming: A case study using the steam platform. In *2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*. IEEE, 1–6.

[63] Karina Sokolova, Charles Perez, and Marc Lemercier. 2017. Android application classification and anomaly detection with graph-based permission patterns. *Decision Support Systems* 93 (2017), 62–76.

[64] Gerard Steen. 1999. Genres of discourse and the definition of literature. *Discourse Processes* 28, 2 (1999), 109–120.

[65] John Swales. 1990. *Genre analysis: English in academic and research settings*. Cambridge University Press.

[66] Juan J Vargas-Iglesias. 2018. Making sense of genre: The logic of video game genre organization. *Games and Culture* (2018), 1555412017751803.

[67] Csaba Veres. 2006. The language of folksonomies: What tags reveal about user classification. In *International Conference on Application of Natural Language to Information Systems*. Springer, 58–69.
[68] Margaret Wallace and Brian Robbins. 2006. Casual games white paper. *IGDA Casual Games SIG, http://www. igda. org/casual/IGDA_CasualGames_Whitepaper_2006. pdf (accessed April 9, 2008)* (2006).
[69] Haoran Wen, EA Leicht, and Raissa M D'Souza. 2011. Improving community detection in networks by targeted node removal. *Physical Review E* 83, 1 (2011), 016114.
[70] Zach Whalen. 2004. Game/genre: A critique of generic formulas in video games in the context of "the real". *Works and Days* 22, 43/44 (2004), 289–303.
[71] Travis W Windleharth, Jacob Jett, Marc Schmalz, and Jin Ha Lee. 2016. Full steam ahead: A conceptual analysis of user-supplied tags on Steam. *Cataloging & Classification Quarterly* 54, 7 (2016), 418–441.
[72] Ludwig Wittgenstein. 1953. Philosophical investigations. Philosophische Untersuchungen. (1953).
[73] Mark JP Wolf. 2001. Genre and the video game. *The medium of the video game* (2001), 113–134.
[74] Changsheng Xu, Namunu C Maddage, Xi Shao, Fang Cao, and Qi Tian. 2003. Musical genre classification using support vector machines. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, Vol. 5. IEEE, V–429.
[75] Hsin-Chang Yang and Zi-Rui Huang. 2018. Mining personality traits from social messages for game recommender systems. *Knowledge-Based Systems* (2018).
[76] Junzhou Zhao, John C.s Lui, Don Towsley, and Xiaohong Guan. 2014. Measuring and maximizing group closeness centrality over disk-resident graphs. 689–694. https://doi.org/10.1145/2567948.2579356
[77] Howard Zhou, Tucker Hermans, Asmita V Karandikar, and James M Rehg. 2010. Movie genre classification via scene categorization. In *Proceedings of the 18th ACM international conference on Multimedia*. ACM, 747–750.
[78] Chengsheng Zhu, Tom O Delmont, Timothy M Vogel, and Yana Bromberg. 2015. Functional basis of microorganism classification. *PLoS computational biology* 11, 8 (2015), e1004472.

# PUBLICATION VII

**A Data-Driven Approach for Video Game Playability Analysis Based on Players' Reviews**

X. Li, Z. Zhang and K. Stefanidis

*Information* 12.3 (2021), 129

*information*

MDPI

*Article*

# A Data-Driven Approach for Video Game Playability Analysis Based on Players' Reviews

**Xiaozhou Li *, Zheying Zhang and Kostas Stefanidis**

Faculty of Information Technology and Communication Sciences, Tampere University, Kalevantie 4, 33100 Tampere, Finland; zheying.zhang@tuni.fi (Z.Z.); konstantinos.stefanidis@tuni.fi (K.S.)
* Correspondence: xiaozhou.li@tuni.fi

**Abstract:** Playability is a key concept in game studies defining the overall quality of video games. Although its definition and frameworks are widely studied, methods to analyze and evaluate the playability of video games are still limited. Using heuristics for playability evaluation has long been the mainstream with its usefulness in detecting playability issues during game development well acknowledged. However, such a method falls short in evaluating the overall playability of video games as published software products and understanding the genuine needs of players. Thus, this paper proposes an approach to analyze the playability of video games by mining a large number of players' opinions from their reviews. Guided by the game-as-system definition of playability, the approach is a data mining pipeline where sentiment analysis, binary classification, multi-label text classification, and topic modeling are sequentially performed. We also conducted a case study on a particular video game product with its 99,993 player reviews on the Steam platform. The results show that such a review-data-driven method can effectively evaluate the perceived quality of video games and enumerate their merits and defects in terms of playability.

**Keywords:** playability; player reviews; text classification; sentiment analysis; topic modeling; steam



**Citation:** Li, X.; Zhang, Z.; Stefanidis, K. A Data-Driven Approach for Video Game Playability Analysis Based on Players' Reviews. *Information* **2021**, *12*, 129. https://doi.org/10.3390/info12030129

Academic Editor: Vincenzo Moscato

Received: 27 February 2021
Accepted: 15 March 2021
Published: 17 March 2021



## 1. Introduction

Playability has been widely acknowledged as the key concept reflecting the overall quality of a video game, in terms of its rules, mechanics, goals, and design within the process of design and analysis [1]. This concept is commonly used in game studies. It reflects the players' degree of satisfaction towards their various ways of interaction with the game system, that is, in a nutshell, *"A good game has good playability"* [2]. It can also be narrowly interpreted as being equal to the quality of "gameplay" or simply the usability of video games, that cannot be balanced by "any non-functional designs" [3]. It is also common to consider both "gameplay" and "usability" as parallel elements of the playability framework [4,5]. Moreover, playability is also seen as the quality in use [6] of video games and represents *"the degree in which specific player achieve specific game goals with effectiveness, efficiency, flexibility, security and, especially, satisfaction in a playable context of use"* [7]. Thus, seeing games as systems and taking into account also the technical, mechanical, or material quality of video games, playability is *"the design quality of a game, formed by its functionality, usability, and gameplay"* [8].

Scholars across domains agree that playability, however measured, can be used to reflect and evaluate the quality of a video game [1,8]. However, regardless of the definition or framework adopted, research on the approaches to analyze the playability of a particular game is limited. The most commonly adopted approach to playability analysis is the use of heuristics [4,5,9,10]. Acknowledgedly, heuristic evaluation has multiple advantages including being cheap, being easy to motivate evaluators, not requiring advanced planning, and importantly it can be done in the early development stage [11]. However, it is also inevitable that such evaluation is biased by the mindset of the evaluators [11,12]. Their experiences and preferences influence the outcome as well [9,13–17]. In addition, it is

common that such usability test contains inconsistency due to the evaluator effect [9,18]. Furthermore, the difference in game rule structures, i.e. games of emergence and games of progression [19], has not been considered in playability evaluation using heuristics. It is obviously not possible to detect gameplay issues of the game elements appearing in the later game scenes of progression games (e.g., Witcher 3 [20]) with the limited time spent on testing with game demos [10].

Hence, towards relevantly fair evaluation on the overall playability of any released video game product, the opinions of the players who have played it for a fair amount of time shall be valuable. Players' game reviews are then the target data source for such purposes. For software products, the analysis of end user reviews has been considered important towards evaluation of software quality [21,22]. Meanwhile, text and opinion mining is a well-known way of *"using large text collections to discover new facts"* [23,24]. With such support, many studies have provided various approaches towards effective review analysis to uncover the critical user needs for software products [25–27]. Despite the differences between video games and utilitarian software products and in the review styles, such players reviews can be used towards the improvement of game products [28]. As one of the most popular digital game distribution platforms, Steam (https://store.steampowered.com/, accessed on 16 March 2021) provides an online venue for the players to review games. With such a large amount of players' opinion data at hand and together with the opinion mining techniques, it is then possible to evaluate video game playability from the perspective of players' collective intelligence.

Herein, we propose a data-driven video game playability analysis approach based on the collection of player textual reviews. It answers the following research question: *How can the data-driven approach be used to gain insights into the playability of a game?* The proposed method uses a pre-trained text classifier model to elicit informative reviews from the pre-processed review collection and uses another pre-trained classifier to classify such reviews into pre-defined playability categories. In this paper, we choooe Paavilainen's game-as-system definition of playability as the reference of classification [8]. With such an explicit and simplified framework and the proposed method, we can obtain not only the intuitively quantified evaluation of the overall playability of the target game but also the specified merits and defects of it in every framework-oriented perspective (answering the research question). We also validated the usefulness of the proposed approach by conducting a case study on a real-life video game with 99,993 reviews.

Compared to heuristic evaluation on playability, this approach relies on the collective intelligence of a large number of players instead of a few experts' opinions. Furthermore, this approach evaluates the game by its released versions instead of demos. Thus, it can provide both the overall playability evaluation and the detailed merits and defects on a game-as-system level. Therefore, although acknowledging the usefulness of heuristic evaluation in game development, we emphasize that the contributions of our approach are: (1) to help game developers obtain a quick overall impression of the perceived game playability from players' perspective; and (2) to help game developers understand the collective needs and complaints of players to identify the playability issues for video game maintenance and evolution.

The remainder of this paper is organized as follows. Section 2 introduces the related work. Section 3 presents the playability analysis approach, including the series of procedures and details. Section 4 presents the case study on validating the proposed approach. Section 5 provides further discussion. Section 6 concludes the article.

## 2. Related Work

### *2.1. Playability Evaluation with Heuristics*

Heuristic evaluation, targeting originally usability evaluation, is an informal analysis method where several evaluators are asked to comment on the target design based on predefined heuristics/principles [11,29]. It aims at finding the usability problems during the iterative design process so that such problems can be addressed before software products

releasing [30]. Despite the rapid development of the video game industry, the methodologies for evaluating game quality and player experience are still limited. Therein, heuristic evaluation is still an effective way of evaluating games compared to other methods for being cheap and fast [31].

Malone proposed the set of heuristics as a checklist of ideas to be considered for designing enjoyable user interfaces which is largely seen as the earliest game heuristics [32]. Therein, three main features are proposed: challenge, fantasy, and curiosity. Federoff's list of game heuristics is based on the observation and interviews with five people from one game development company [33]. For such heuristics, game interface, game mechanics, and game play are the three main aspects. Neither of these early studies provides validation of the respectively proposed heuristics.

Desurvire et al. introduced Heuristic Evaluation for Playability (HEP) towards video, computer, and board game evaluation with four categories: game play, game story, game mechanics, and game usability [4]. HEP is validated via comparison with a user study of a new game at the beginning of its development with four prospective users in 2-h sessions. The authors also emphasized HEP is helpful at the early stage of game design but admitted players' behavior is still unpredictable. In addition, HEP is extended into Game Genre-Specific Principles for Game Playability (PLAY) to adapt usability principles to game design [34]. Forty-eight game design principles from eight categories are proposed.

Korhonen and Koivisto's playability heuristics are designed for mobile games where gameplay, game usability, and mobility are the main categories [5]. It is validated by four experts over a mobile game in the production phase. The authors also admitted that, although heuristics are helpful, the gameplay is much harder to evaluate. Furthermore, they extended the heuristics to mobile multiplayer games with experiments showing the heuristics can be applied to non-mobile games as well [35]. Korhonen et al. also compared their heuristics with HEP finding the respective strength and weakness [9]. The study also detects inconsistency within evaluators in terms of their reported problems due to the potential evaluator effects [18] or different reporting baseline.

Pinelle et al. proposed heuristic evaluation focusing on the usability for video game design based on the analysis of game expert reviews [10]. The heuristic set contains 12 problem categories and 285 individual problems. It is verified via a testing evaluation of the demo of a PC game by five expert evaluators. Thereafter, an extension study is conducted towards heuristics for networked multiplayer games; as a result, five problem categories with 187 problems specially for network multiplayer games are proposed and verified by 10 expert evaluators on two network games [13]. However, Pinelle and colleagues also emphasized "*the heuristics do not address design issues related to how fun and engaging games are for users*".

Koeffel et al. proposed a three-aspect heuristic set (including game play, game story, and virtual interface) to evaluate the user experience in video games and tabletop games [36]. The authors summarized 29 heuristic items based on extensive literature search and verified the heuristics based on expert evaluation (two experts) on five games of different genres and comparison to game media reviews.

Many other scholars also conduct research on utilizing heuristic evaluation for specific types of games. Röcker and Haar showed that existing heuristics can be transferable to pervasive gaming applications [37]. Tan et al. proposed using heuristic evaluation, the Instructional Game Evaluation Framework, for educational game analysis [38]. Khanana and Law illustrated the use of playability heuristics as design tools for children's games [39]. However, whether these heuristics can be used for video games in general is not verified.

On the other hand, regarding the different ways of using heuristic evaluation towards video game playability, Aker et al. found, based on an extensive literature search, that empirical evaluation, expert evaluation, inspection, and mixed method are the methods used for such purpose [40]. Among the four mentioned, expert evaluation is the most commonly applied with many of the above mentioned studies adopting such a method [5,9,13,33,35,36]. However, the outcomes of such a method rely heavily on the

experts' skills and preferences and seldom capture the behaviors and needs of real end users [41]. Empirical evaluation, such as surveys, interviews, and focus groups, is also a relevantly common method of using heuristics [4,32,37,38]. However, with such a method, it is difficult to properly select the correct user sample and reproduce actual usage/play situations within the limited given time [41].

### *2.2. User Review Studies*

Being an important data source, customer feedback is commonly used for companies to understand the market and the needs of their customers so that they could improve products and services accordingly. Regarding software products, it is also critical to facilitate the evolution of software products and services via the analysis of end user reviews [21,22].

Many studies show mining the end user reviews of software products can help reveal the hidden user behaviors, software characteristics, and the relations in between. Vasa et al. conducted a preliminary analysis on 8.7 million reviews of 17,330 mobile apps using statistic methods on user review character counts and ratings [42]. Their results show mobile app reviews tend to be short and both the rating and the category of an app influence the length of a review. With the same data, Hoon et al. showed that the most frequently used words in user reviews are to express sentiment [43]. Harman et al. used customized algorithms to extract app features and correlation analysis on 32,108 non-zero priced apps from Blackberry app store [44]. The results show a strong correlation between customer rating and the app download ranking but no correlation between the app price and either downloads or ratings.

More importantly, many studies also show that the results from mining end user reviews can reflect the positive and negative user experience regarding software products. For example, Vu et al. proposed a keyword-based review analysis method to detect keyword trends and sudden changes that could possibly indicate severe issues [45]. Panichella et al. proposed an approach to extract information from user reviews relevant to the maintenance and evolution of mobile apps using Natural Language Processing (NLP), sentiment analysis, and text analysis techniques [46]. Gu and Kim proposed a method to categorize reviews, extract aspects from sentences, and evaluate the obtained aspects of the mobile apps using NLP techniques [47]. Many other studies also show that opinion mining on end user reviews can help identify user complaints [48], the useful information [26], and the factors for software success [25] and evaluate the experience towards specific software features [49], merits and defects of particular software updates [50,51], and software evolution [27].

Despite the differences in video games and utilitarian software products, as well as those between the review styles, such end user reviews are considered valuable for game designers and developers towards the improvement of their game products. Lin et al. conducted an empirical study on the reviews of 6224 games on Steam and analyzed the review content and the relation between players' play hours and their reviews [28]. Santos et al. compared the expert and amateur game reviews on Metacritic and found amateur reviews are more polarized and have stronger sentiments than expert reviews [52]. Lu et al. used topic modeling on Steam reviews to investigate the temporal dynamics of player review topics and the influence of updates to such dynamics [53]. Although game reviews form a rich resource for understanding the players' experience and opinion on a particular game, the game playability analysis based on players' reviews is yet under-explored.

## 3. Method

In this section, we present an overview of our approach towards video game playability evaluation, with further explanation of the key steps.

### *3.1. The Framework Overview*

Figure 1 depicts the overview of the video game playability evaluation approach with each key step specified. The framework starts with collecting the player reviews from online platforms (e.g., Steam) via either API or web crawling. Then, we preprocess the obtained raw review data into structured form. The second step is to filter out the "non-playability-informative" reviews via a pre-trained classifier. With the obtained "playability-informative" reviews dataset, the third step is to classify the data into different playability perspectives according to a selected playability framework. For example, when selecting the framework of Paavilainen [8], the reviews are then classified into three perspectives accordingly, i.e., functionality, gameplay, and usability. With each review instance categorized into a specific perspective, the fourth step is to quantify the evaluation result of each perspective. Subsequently, the fifth step is to visualize such a result and present an intuitive summary. Meanwhile, the sixth step is to extract the existing merits and defects from each perspective by modeling and summarizing the topics of the reviews within. The output of both the visualization and topic modeling is then synthesized into a report.

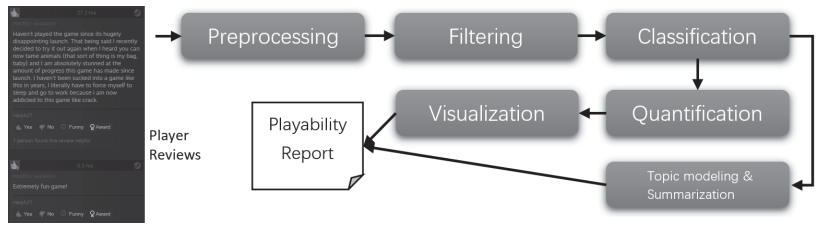


**Figure 1.** Video Game Playability Evaluation Framework.

### *3.2. Preprocessing*

The preprocessing step encompasses the following key activities. First, we divide each review item from the dataset into sentence-level review instances, due to the fact that each review with multiple sentences can contain multiple topics and various sentiments. In this study, we use the sentence tokenizer feature from the Natural Language ToolKit (NLTK) (http://www.nltk.org/, accessed on 16 March 2021), a popular Python package with text processing libraries, corpora, and lexical resources. Secondly, based on the obtained sentence-level review dataset, we build the bigram and trigram models to identify the phrases within the data. For such a purpose, we use the phrase detection feature of Gensim (https://radimrehurek.com/gensim/models/phrases.html, accessed on 16 March 2021), a popular semantic modeling package. Subsequently, for each review sentence, we perform a series of text processing activities, including transforming text into lowercase, removing non-alpha-numeric symbols, screening stop-words, eliminating extra white spaces, and lemmatization (using the WordNetLemmatizer model of NLTK). Note that the processing is only applied to the text when topic modeling is required. For sentiment analysis, such activities are not only unnecessary but also counter-productive.

### *3.3. Filtering*

Herein, the filtering step is to classify the dataset of sentence-level review instances into "playability-informative" and "non-playability-informative". By doing so, we identify the review sentences that contain description regarding the playability of the particular game and screen out those not relevant. Due to the variety in playability definition, the criteria by which review instances are categorized slightly vary. In this study, we adopt the game-as-system playability definition given by Paavilainen [8] as a reference, as this definition provides clear criteria for the identification of playability-related text with a pre-defined playability perspective framework with minimum complexity compared to the other frameworks. Thus, accordingly, we set the two unique class labels as {'Playability-informative (P)', 'Non-Playability-Informative (N)'}. Based on the adopted definition and

framework, the criteria for "playability-informative" reviews are listed in Table 1 with explanation and examples attached. On the other hand, a review is accordingly labeled "non-playability-informative" when it contains no information related to such criteria. For example, review sentences such as "*I'm glad I supported this Dev team.*" (Development and Publishing), "*Now the game has exceeded my expectations!*" (Feeling Expression), and "*Ive got this game on PS4, XBOX and now Steam.*" (Player Self Description) are all seen as "non-playability-informative".

**Table 1.** "Playability-informative" Criteria based on Paavilainen's Framework [8] and Examples.

| Criteria | Explanation | Review Examples |
| --- | --- | --- |
| **Functionality** | the technical, mechanical or material quality of the game that is related to its smooth operation. | *"...the performance in VR mode is absolutely terrible."* *"Crashing and stuttering constantly..."* |
| **Gameplay** | the rule dynamics that provide "gameness". e.g., goals, challenge, progress, and rewards. | *"Survival is not challenging unless you play hardcore,..."* *"...doing the same repetitive things over and over again"* |
| **Usability** | the user-interface of the game and its ease of use. | *"Controls and menus are bad,..."* *"...the massive improvements to the games graphics..."* |

To efficiently identify and filter the "non-playability-informative" review sentences, we herein apply a classifier based on machine learning algorithm. In the study, we compare the Naive Bayes (NB) and the Expectation Maximization for Naive Bayes (EMNB) [54] and adopt the EMNB classifier in the filtering step. EMNB is a well-recognized semi-supervised text classification algorithm, which can build a classifier with high accuracy using only a small amount of manually labeled training data. With EMNB, we thus filter out the review sentences labeled 'N' and build the "playability-informative" review sentence dataset.

### *3.4. Classification*

In this step, we classify the obtained "playability-informative" review sentences into perspectives according to the selected playability framework. As stated above, in this study, we adopt the playability framework that contains three perspectives, i.e., Functionality (F), Gameplay (G), and Usability (U). Targeting the specific objectives of this study when the classes (i.e., playability perspectives) are determined by the existing framework, a supervised learning algorithm is more suitable. On the other hand, it is also frequent that a particular review sentence contains information regarding multiple perspectives. For example, the review sentence "*The gameplay, UI and story are not bad, unfortunately this game has no Beginner friendly and you had to figure out by your own.*" describes the players' opinion on both gameplay and usability. Thus, to cope with such a situation, we adopt a multi-label classification algorithm. For such a multi-label classification task, we select from three algorithms: kNN classification method adapted for multi-label classification (MLkNN) [55], Twin multi-Label Support Vector Machines (MLTSVM) [56], and Binary Relevance multi-label classifier based on k-Nearest Neighbors method (BRkNN) [57]. The interfaces of these classification algorithms are provided by the Scikit-multilearn (http://scikit.ml, accessed on 16 March 2021), a BSD-licensed library for multi-label classification built on top of the Scikit-learn ecosystem (https://scikit-learn.org/, accessed on 16 March 2021). The comparison of these algorithms is discussed in Section 4.2.

### *3.5. Quantification*

In this step, with the classified three sets of "playability-informative" review sentences, we evaluate each of the playability perspectives by quantifying the overall opinions extracted from the according set of review sentences.

Herein, we use the average sentiment score of the "playability-informative" review sentences representing the players' collective evaluation towards the playability of the game.

Algorithm 1 depicts the process of quantifying the playability of a particular game. Let $R$ be the set of "playability-informative" review sentences, where each $r_i \in R$ is evaluated

with the sentiment score ($s_i$) assigned via a selected sentiment analysis method, e.g., Valence Aware Dictionary for sEntiment Reasoning (VADER) [58], Sentiment strength [59], etc. Meanwhile, each $r_i \in R$ is labeled by one or more playability perspectives ($L_i$) using the pre-trained multi-label text classifier (i.e., MLclassifier). Thereafter, for each playability perspective ($p$), we find the set $R_p$ that contains all the review sentences labeled $p$ and calculate the sentiment value for such perspective as the average of the sum of the sentiment score (see Line 10).

**Algorithm 1:** Algorithm of Quantifying the Playability on Multiple Perspectives.

**Data:** A set of "playability-informative" review sentences
**Result:** A dictionary of playability scores, each for one perspective
1 $R \leftarrow$ set of "playability-informative" review sentences;
2 Let $P$ be the set of all playability perspective labels;
3 **for** ***each*** $r_i \in R$ **do**
4 $\quad s_i \ (\in S) \leftarrow$ getSentimentScore($r_i$);
5 $\quad L_i \ (\subseteq P, \neq \emptyset) \leftarrow$ MLclassifier.predict($r_i$);
6 **end**
7 Let *result* be return dictionary;
8 **for** ***each*** $p \in P$ **do**
9 $\quad R_p \leftarrow$ any $r_i$ has $p \in L_i$;
10 $\quad v_p \leftarrow \frac{\sum_{r_i \in R_p} s_i}{len(R_p)}$;
11 $\quad result[p] \leftarrow v_p$;
12 **end**
13 return *result*;

### *3.6. Visualization*

In this step, we visualize the output of the quantification of player opinions regarding each playability perspective with a polygon diagram. The number of vertices of the selected polygon is equal to the perspective numbers. For example, when adopting Paavilainen's playability framework of three perspectives, the analysis of playability to a particular game can be depicted as a triangle chart (Figure 2).

As shown in Figure 2, the line segments from the triangle center to each vertex represent the scales measuring each playability perspective. The distance between a green playability triangle vertex and the center represents the playability score in the particular perspective. The central point of each line segment is value *0* indicating the neutrality of the according perspective. The larger the green triangle is, the higher the overall playability score the game has. When the red area is shown in any direction, the playability of that game suffers in that particular perspective. Herein, the scale range indicating the positive and negative of each perspective is determined by the average sentiment score ranging from $-1$ to 1. For this particular game example, after the analysis of its reviews through Algorithm 1, a result list $[-0.2, 0.5, 0]$ is obtained. Based on such a result, Figure 2 is drawn. It shows the game is good for its gameplay ($G = 0.5$), mediocre for its usability ($U = 0$), and unsatisfactory for its functionality ($F = -0.2$).

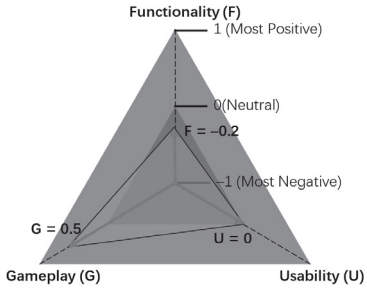


**Figure 2.** An example of playability triangular chart.

### *3.7. Topic Modeling and Summarization*

As a critical part of the playability evaluation results, finding the players' opinions regarding the pros and cons of a particular game product can convey great value to the developer team. Thus, in this step, besides the quantified outcome of overall playability, we identify the specific issues regarding each playability perspective using a topic modeling algorithm. In this study, we use Latent Dirichlet allocation (LDA) topic modeling, a well-recognized effective topic modeling algorithm that finds the hidden topics from a large set of text data [60].

## 4. Case Study

In this section, we verify the effectiveness of the proposed playability evaluation method by conducting an experiment on a real-life video game from Steam platform.

### *4.1. Data Description*

The game we select for this case study is No Man's Sky (NMS) [61], a space exploration and survival game developed and published by Hello Games (https://hellogames.org/, accessed on 16 March 2021). The game was first released on 12 August 2016, before which a social media "hype" had been evoked leading to an unprecedentedly high expectation from the players. However, the release of the game was disastrous due to the unfulfilled promises from the developers as well as the performance and gameplay defects. Interestingly, for the last four years, the game has been continuously maintained with its quality gradually increasing, which makes it a unique case where the changes in game quality is observable.

We collected the 99,993 English reviews from 12 August 2016 to 7 June 2020 for NMS. Within the collected review set, the longest contains 116 sentences while the shortest is a single-sentence review. Via tokenization, we obtained 519,195 review sentences. We then manually labeled the sentences with "Playability-informative" (P) and "Non-playability-informative" (N) in a random order, until obtaining 1500 "playability-informative" sentences and 1500 "non-playability-informative" sentences. Therein, 1000 sentences (500 for each label) were saved as training data and 2000 (1000 for each label) as testing data for building the filtering classifier model. Furthermore, adopting Paavilainen's playability framework, we further labeled the 1500 "playability-informative" review sentences in both the training and testing dataset into *Functionality (F)*, *Gameplay (G)*, and *Usability (U)*, where it is possible for one sentence to contain multiple labels. Such dataset was used to train the classifying model. Note that the labeling of the training data is ideally done by three expert evaluators. Two evaluators first label the sentences separately and then each label is confirmed by the agreement of both parties. A third evaluator is invited to provide final verification when agreement cannot be reached.

### *4.2. Classifier Selection*

To evaluate the performance of the proposed method, we conducted experiments testing its key steps, including the filtering and the classification steps.

#### 4.2.1. Filtering Evaluation

To evaluate the performance of filtering, with a series of experiments, we compared the results of the original NB algorithm and the EMNB algorithm with the amount of training and test data from 60 to 3000 with a step of 60. Within the amount of data selected for each experiment iteration, 1/3 was selected as training data with the other 2/3 as test data. The evaluation results show that the performances of NB and EMNB are similar regarding our dataset throughout various data volumes. Throughout the series of experiments, the accuracy (F1-score) difference between NB and EMNB with that same data volume does not exceed 0.04. On the other hand, with a limited number of training data (100 training data and 200 testing data), the accuracy of both algorithms reaches a satisfactory level ($\geqslant$0.7). The level of accuracy does not drop when enlarging the data volume. Furthermore, with the data volume reaches around 1200, both classification algorithms can provide optimal accuracy ($\geqslant$0.8). In this study, considering the large amount of unlabeled review sentence data as well as the according efficiency, we adopted the EMNB algorithm with the full training data volume in order to obtain the best performance (F1-Score = 0.85).

#### 4.2.2. Classification Evaluation

Furthermore, we conducted a series of experiments to compare the performances of three multi-label text classification (MLTC) algorithms, i.e., MLkNN, MLTSVM, and the two versions of BRkNN. With the manually labeled 1500 "playability-informative" training data, we first found the best parameters targeting the best performance for each algorithm. Then, the best accuracy of the three algorithms with the detected parameters were calculated for comparison. The results shown in Table 2 indicate that MLkNN algorithm has the best classification accuracy (0.769) on our training dataset with the detected best parameter. Together with the previous filtering step with EMNB (accuracy of 0.85), the overall accuracy is satisfactory ($0.85 * 0.769 = 0.653$).

**Table 2.** Comparison of the performance of MLTC algorithms.

| Algorithm | Best Parameter | Accuracy |
|---|---|---|
| MLkNN | k = 27, s = 0.5 | 0.769 |
| MLTSVM | c_k = 0.125 | 0.532 |
| BRkNNaC | k = 19 | 0.663 |
| BRkNNbC | K = 17 | 0.712 |

In addition, to further tune the method, we evaluated both the performance of combining the two individual steps and that of applying only the MLTC algorithm targeting both filtering and classifying tasks. For such purpose, we manually labeled "N" to the 1500 "non-playability-informative" training data and combined them with the 1500 "playability-informative" ones. The performance of the above three algorithms on the enlarged dataset is shown in Table 3.

**Table 3.** Comparison of the performance of two- and one-step classification.

| Two-Step | | | One-Step | | |
|---|---|---|---|---|---|
| **Algorithm** | **Best Parameter** | **Accuracy** | **Algorithm** | **Best Parameter** | **Accuracy** |
| EMNB + MLkNN | k = 27, s = 0.5 | 0.653 | MLkNN | k = 1, s = 0.5 | 0.121 |
| EMNB + MLTSVM | c_k = 0.125 | 0.452 | MLTSVM | c_k = 0.125 | 0.349 |
| EMNB + BRkNNaC | k = 19 | 0.564 | BRkNNaC | k = 1 | 0.121 |
| EMNB + BRkNNbC | K = 17 | 0.605 | BRkNNbC | k = 6 | 0.276 |

The results show that a two-step classification, i.e., "playability-informative" review filtering with EMNB and perspective classifying with multi-label text classification, has a much better accuracy rate than one-step classification with only MLTC. In addition, we

found that using MLkNN (with the parameter $k = 27$ and $s = 0.5$) for the classifying procedure has the best overall accuracy.

### *4.3. Results*

In this section, we present the results from applying the proposed playability analysis approach on the review dataset of NMS. The results contain two major parts: (1) the overall playability score; and (2) the merits and defects of the game.

#### 4.3.1. Playability Score

With the obtained 519,195 review sentence data, we follow the analysis method procedure by first filtering out the "non-playability-informative" ones. As an outcome, 273,476 review sentences are automatically labeled as "playability-informative" using the pre-trained EMNB classifier with the 3000 training data. Subsequently, we classify them into the three perspectives using the selected MLkNN algorithm receiving 43,110 review sentences on functionality, 20,5474 on gameplay, and 30,176 on usability. To perform sentiment analysis on the review sentences, we select the VADER approach, due to its high classification accuracy on sentiment towards positive, negative, and neutral classes in social media domain [58]. In addition, its overall classification accuracy on product reviews from Amazon, movie reviews, and editorials also outperform other sentiment analysis approaches and run closely with that of an individual human [58]. It is also easy to import and perform using Python as being integrated into the NLTK package. By calculating the sentiment score for each review sentence with VADER and the average score for each review set, we obtain the result as {'Functionality': 0.025, 'Gameplay': 0.111, 'Usability': 0.039} (shown in Figure 3). It indicates that the overall playability of this game is at the level of mediocre in each of the two out of three perspectives, when only performs only slightly better than mediocre in the gameplay perspective. Such results comply with the overall rating of *Mixed* on Steam (https://store.steampowered.com/app/275850/No_Mans_Sky/#app_reviews_hash, accessed on 16 March 2021).

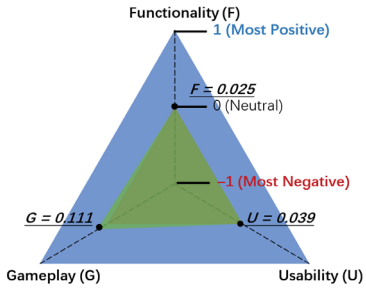


**Figure 3.** Overall playability score for NMS.

Furthermore, to further verify the results, we follow the major updates of NMS via the information from its patch notes (https://nomanssky.gamepedia.com/Patch_notes, accessed on 16 March 2021). As the release date of the 11th major update is 11th June 2020, the review dataset for the 10th update is incomplete. Thus, focusing on the first nine updates (*Foundation*, *PathFinder*, *Atlas Rises*, *NEXT*, *Abyss*, *Visions*, *Beyond*, *Synthesis*, and *Living Ship*), we divide the "playability-informative" review sentences into 10 subsets based on their release dates. Via the same calculation on each subset, the playability analysis results regarding the original release (marked as Release 1.0) and the nine following major updates, as well as their data volumes, are shown in Table 4.

**Table 4.** Playability score changes through major updates.

| | 1.0 | Foundation | PathFinder | Atlas Rises | NEXT | Abyss | Visions | Beyond | Synthesis | Living Ship |
|---|---|---|---|---|---|---|---|---|---|---|
| **Date** | 16.11.27 | 17.03.08 | 17.08.11 | 18.07.24 | 18.10.29 | 18.11.21 | 19.08.14 | 19.11.28 | 20.02.18 | 20.04.07 |
| **Count F.** | 28,251 | 800 | 718 | 1851 | 4222 | 135 | 1858 | 2680 | 1052 | 555 |
| **Count G.** | 120,894 | 6128 | 5460 | 12,582 | 19,085 | 554 | 12,649 | 10,011 | 7667 | 3579 |
| **Count U.** | 18,106 | 758 | 751 | 1698 | 2876 | 103 | 1692 | 1900 | 922 | 449 |
| **Score F.** | −0.0054 | 0.0372 | 0.0973 | 0.0684 | 0.0487 | 0.0091 | 0.0760 | 0.0458 | 0.0966 | 0.0770 |
| **Score G.** | 0.0765 | 0.1322 | 0.1346 | 0.1534 | 0.1389 | 0.1080 | 0.1686 | 0.1406 | 0.2211 | 0.2113 |
| **Score U.** | 0.0106 | 0.0708 | 0.0807 | 0.0948 | 0.0608 | 0.0823 | 0.0755 | 0.0481 | 0.1489 | 0.1538 |

Based on such results, we can conclude that the playability of the game increased in terms of all three perspectives through the nine updates, even though it decreased regarding some particular updates (e.g., Beyond). The reason for such a situation is the introduction of new critical features, major interface changes, new vital bugs, etc. Taking the Beyond update as an example, as a Version 2.0.0, it added the Virtual Reality support and a wide range of features to the game (https://nomanssky.gamepedia.com/Update_2.00, accessed on 16 March 2021). It evoked controversy among players regarding its performance and gameplay. Nonetheless, by comparing the playability of Release 1.0 and that of the version after the "Living Ship" update, all three perspectives had been greatly improved.

#### 4.3.2. Playability Merits and Defects

To detect the merits and defects of the game in terms of each playability perspective, we first divide the review sentences into three subsets based on the classification result. For each subset, i.e., the review sentences for each perspective, we further select the positive (sentiment score greater than 0) review sentences and the negative ones (sentiment score smaller than 0) forming six review sentence sets. The volume of each subset is shown in Table 5. To detect the explicit sentiment from the review, we ignore the neutral (sentiment score equals 0) review sentences herein. In addition, to conveniently compare the results to the information extracted from the Metacritic later, we select only the review data concerning the original game release (i.e., between 12 August 2016 and 27 November 2016).

**Table 5.** Data volume for review subsets for Release 1.0.

| | Functionality | Gameplay | Usability |
|---|---|---|---|
| **Positive** | 10,684 | 47,780 | 6537 |
| **Negative** | 10,637 | 32,627 | 6396 |

Subsequently, to find the best topic number for each review subset, we conduct a series of experiments for each set testing with the topic numbers ranging from 2 to 20. We use the topic coherence representing the quality of the topic models. Topic coherence measures the degree of semantic similarity between high scoring words in the topic. A high coherence score for a topic model indicates the detected topics are more interpretable. Thus, by finding the highest topic coherence score, we can decide the most fitting topic number. Herein, we use $c\_v$ coherence measure, which is based on a sliding window, one-set segmentation of the top words and an indirect confirmation measure that uses normalized pointwise mutual information (NPMI) and the cosine similarity [62]. Note that we pick the model that has the highest $c\_v$ value before flattening out or a major drop, in order to prevent the model from over-fitting.

With the best topic number ($k$) values detected for the six review subsets (shown in Table 6), we can continue with building the according topic models and detecting the topics. Table 7 shows the extracted topics for each subset as well as the top 10 keywords that describe each of them.

**Table 6.** Best fitting topic numbers and c_v values.

| | **Functionality** | **Gameplay** | **Usability** |
|---|---|---|---|
| **P** | $k = 3, c\_v = 0.552$ | $k = 4, c\_v = 0.535$ | $k = 3, c\_v = 0.397$ |
| **N** | $k = 3, c\_v = 0.438$ | $k = 3, c\_v = 0.522$ | $k = 5, c\_v = 0.374$ |

**Table 7.** Detected topics for each review set.

| **Topic (Positive Functionality)** | **Top Words** |
|---|---|
| **+ Load Screen and Crashing** | "game", "play", "crash", "time", "screen", "start", "hour", "go", "load", "get" |
| **+ Performance and bugs fixed via update** | "issue", "game", "performance", "fix", "people", "update", "would", "bug", "problem", "patch" |
| **+ Running game fine with settings** | "run", "game", "setting", "graphic", "work", "get", "fps", "pc", "fine", "high" |
| **Topic (Negative Functionality)** | **Top Words** |
| **− Poor Performance, Bugs, Crash, Need Fix** | "game", "issue", "problem", "performance", "fix", "people", "crash", "bad", "poor", "bug" |
| **− Lag, Stutter, fps drop, even with low settings** | "run", "setting", "low", "game", "stutter", "drop", "pc", "graphic", "lag", "fps" |
| **− Crash at Start screen, try hours** | "crash", "game", "play", "time", "screen", "get", "start", "can", "try", "hour" |
| **Topic (Positive Gameplay)** | **Top Words** |
| **+ Explore, survival, different planet systems** | "planet", "find", "new", "explore", "system", "beautiful", "different", "look", "survival", "thing" |
| **+ Crafting, ship-flying, resource and inventory** | "space", "ship", "get","resource", "fly", "well", "craft", "upgrade", "inventory", "learn" |
| **+ Fun exploration gameplay** | "game", "exploration", "fun", "play", "get", "hour", "gameplay", "good", "enjoy", "lot" |
| **+ Need story to make better** | "game", "want", "make", "need", "would", "give", "bit", "people", "story", "work" |
| **Topic (Negative Gameplay)** | **Top Words** |
| **− Repetitive, boring gameplay** | "game", "get", "hour", "feel", "repetitive", "start", "bore", "boring", "gameplay", "people" |
| **− Lack of inventory upgrade** | "ship", "resource", "make", "need", "inventory", "find", "upgrade", "lack", "craft", "much" |
| **− Fly, explore, combat** | "planet", "space", "see", "look", "explore", "combat", "find", "fly", "kill", "ship" |
| **Topic (Positive Usability)** | **Top Words** |
| **+ Control feels with controller, fly ship** | "control", "use", "ship", "take", "feel", "get", "controller", "fly", "space", "flight" |
| **+ Beautiful graphics** | "game", "graphic", "play", "change", "setting", "beautiful", "look", "run", "work", "good" |
| **+ Music&sound, hold and click button** | "hold", "button",' 'music", "menu", "screen", "system", "inventory", "click", "sound", "second" |
| **Topic (Negative Usability)** | **Top Words** |
| **− Graphic settings poor, restart** | "graphic", "game", "setting", "change", "run", "bad", "start", "poor", "get", "restart" |
| **− Fly control with mouse annoying** | "control", "mouse", "ship", "fly", "game", "use", "get", "annoying", "make", "press" |
| **− Terrible texture and sound** | "terrible", "look", "texture",' 'game", "sound", "pop", "point", "require", "complaint", "way" |
| **− Horrible flight control, cluncky inventory** | "control", "flight", "feel", "people", "horrible", "system", "inventory", "lack", "clunky", "fov" |
| **− Option, click and hold button, bad/awful PC port** | "game", "pc", "option", "button", "hold", "port", "menu", "awful", "bad", "click" |

From the detected topics, we can easily summarize the merits and defects of the game in terms of each playability perspective. For example, the topics extracted from the "negative-functionality" review set show that users are satisfied with the performance of the game when settings are tackled properly. They are also satisfied with the bugs being fixed and with the game despite the load screen and crashing. On the negative side, players often complain about various issues, including poor performance, bugs, crashes, lagging, stuttering, fps, etc. Regarding gameplay, the exploration and survival through different planet systems, as well as the crafting, spaceship cruising, and resource and inventory management, are well received by the players. They also indicate a better story is needed. On the other hand, the players feel negative about the gameplay being repetitive and boring and frustrated about the lack of inventory upgrade. The flying, exploring, and combat mechanisms also suffer. Regarding usability, the players feel positive regarding the spaceship control using controller and the beautiful graphics. They also like the music and sound effects and the menu interface using a click and hold button to access the inventory. However, players also complain about the following aspects: the graphic setting only changes after restarting, controlling with mouse is annoying, texture and sound being terrible, horrible flight control and clunky inventory, the click-and-hold interaction mechanism, and being an awful PC port. Note that a similar topic shown in both the positive and negative groups (e.g., loading screen and crash) suggests that a relevantly high number of players express different sentiment when talking about 'crash'. For example, *"With a Rift S headset and a gtx1080 graphics card I'm getting great performance out of the game with no crashes."* and *"This game has a TON of performance problems and has crashed on me far*

*too many times to be acceptable."* express sentiment differently when both are about "crash". Such a situation indicates players' opinions diverge regarding this topic.

To verify the correctness of the detected merits and defects of the game via topic modeling, we compare our results to the expert opinions extracted from the critic reviews of Metacritic (https://www.metacritic.com/game/pc/no-mans-sky, accessed on 16 March 2021). Metacritic reviews have been considered valuable in providing insights in evaluating the quality of media products, e.g., movies and games [63,64]. We find the 10 critic reviews (including 'gamewatcher', 'hookedgamers.com', 'ign denmark', 'the games machine', 'mmorpg.com', 'pelit.fi', 'pcgamer.com', 'gamegrin.com', 'games.cz', and 'game-debate.com') on NMS. Their full review contents are accessible online with the "pros and cons" explicitly listed. Due to the fact that all the critic reviews were given soon after the release date, the opinions thus only apply to Release 1.0 of the game. As stated above, such opinions are used to compare with the extracted players' review opinions regarding the same version.

As shown in Table 8, we can easily compare the extracted positive and negative topics from the player reviews and the summarized "playability-informative" "Pros and Cons" from the critic reviews. We can conclude that a great majority of the merits and defects of the game mentioned by the media experts are detected from the player review modeling. For example, regarding functionality, both parties point out the problems of crashing, bugs, frame drops, and performance issues. Note that the critic reviews do not mention the merits regarding functionality, which is reasonable as providing a functional product is clearly a "must-have" instead of an "exciter". Regarding gameplay, the exploration and survival gameplay is praised by both, as well as the different planet systems and spaceship flying. The sense of relaxing that mentioned by the media is not covered by the players' topics. On the negativity of gameplay, the complaints about inventory, repetitive/tedious gameplay (limited options), lack of combat, etc. are mutual. Furthermore, regarding usability, the graphics and sound are praised by both, when the players' reviews additionally give credits to the controlling performance with controllers. On the other hand, both parties reflect negative opinions on the control (with mouse) and menu/option being frustrating, when the players complain more specifically about the "hold and click button" control.

In addition, we also compare these extracted topics to the original game-as-system definition and the according perspective descriptions of Paavilainen's playability framework [8]. Regarding functionality, nearly all the sub-perspectives are covered by the player review topics, except for "error reporting". Apparently, the players are generally not satisfied with functionality from all sub-perspectives, as all such can be related to at least one topic from negative reviews. On the other hand, regarding gameplay, the player reviews reflect positively on the play styles, goals, challenges, and rewards of the game, when convention and consistency are not mentioned enough. Repetitiveness and autonomy (i.e., lack of inventory upgrade -> cannot freely preserve more items) are the gameplay sub-perspectives being complained often. Finally, regarding usability, the negative opinions are about the control with mouse (control), texture (audiovisual), inventory (UI layout), graphic setting, option/menu (Navigation), and click and hold button (feedback). Such results further validate the extracted review topics are "playability-informative".

**Table 8.** Mapping between extracted player review topics and metacritic reviews pros and cons.

| Playability | Players' Review Topic | Metacritic Review Pros and Cons |
|---|---|---|
| **Functionality** | **+ Load Screen and Crashing** | |
| | **+ Performance and bugs fixed via update** | |
| | **+ Running game fine with settings** | |
| | **− Poor Performance, Bugs, Crash, Need Fix** | − Still major technical issues. |
| | | − Deplorable technical condition. |
| | | − The PC version is heavy, buggy, and crashing |
| | **− Lag, Stutter, fps drop, even with low settings** | − Random frame rate drops. |
| | | − Poorly optimized. |
| | **− Crash at Start screen, try hours** | |
| **Gameplay** | **+ Explore, survival, different planet systems** | + Solid survival gameplay with great freedom. |
| | **+ Crafting, ship−flying, resource and inventory** | + Relaxing exploration |
| | **+ Fun exploration gameplay** | + Massive universe to explore. |
| | | + It truly is an impossibly huge galaxy. |
| | | + A sense of majesty and grandeur unlike anything else. |
| | | + Lots of options to fiddle with. |
| | | + Near limitless replay value. |
| | | + Huge scale, infinite content. |
| | | + Solid survival gameplay with great freedom. |
| | | + Relaxing exploration. |
| | **+ Need story to make better** | − Very little real story. |
| | | − No reason to proceed, lacks a narrative... |
| | **− Repetitive, boring gameplay** | − a lack of real discovery |
| | | − Most planets look the same |
| | | − repetitive systems |
| | | − Repetitive |
| | | − Dull, tedious crafting. |
| | | − Planets all hold the same handful of interest points. |
| | | − ... disappoints in almost every way and just has no depth |
| | | − ... gameplay options extremely limited. |
| | | − ... Has too few features to be varied in the long run. |
| | | − soon turns into a routine stereotype... |
| | | − The universe is a lifeless and static backdrop. |
| | **− Lack of inventory upgrade** | − Loads of inventory management. |
| | **− Fly, explore, combat** | − Not for thrill seekers or combat fans. |
| | | − whilst gathering resources to move on but won't linger. |
| **Usability** | **+ Control feels with controller, fly ship** | |
| | **+ Beautiful graphics** | + Beautiful alien worlds. |
| | | + Breathtaking views. |
| | | + Stylish in graphics ... |
| | | + some lovely scenery |
| | | + An atmospheric walk through beautiful worlds |
| | **+ Music and sound, hold and click button** | + stylish..sound |
| | | + A successful.. atmospheric audiovisual implementation. |
| | **− Graphic settings poor, restart** | |
| | **− Fly control with mouse annoying** | |
| | **− Terrible texture and sound** | |
| | **− Horrible flight control FOV, cluncky inventory** | − Uncomfortable controls |
| | **− Option, click and hold button, bad/awful PC port** | − frustrating menus |
| | | + It may work perfectly as an occasional short distraction |
| | | − Many promises left undelivered |

"+" represents positive, "−" represents negative.

## 5. Discussion

Considering that other factors can also influence the outcome of the playability analysis, we extended the experiments using the *playtime* of the players and the *voted helpfulness* value as the weight to the sentiment score. The playtime value indicates how long each player has been playing the game, i.e., the game experience. It is reasonable to assume that players who spend more time on a particular game with more gaming experience shall provide more trustworthy reviews. On the other hand, the voted helpfulness value indicates how many other players agree with the statement and evaluation in a particular review, i.e., the perceived trustworthiness. According to our review data, among the players who wrote the reviews, the longest playtime of one player is 645,618 min (≈10,760 h) with the shortest

being 1 min. The average playtime is 5791.70 min (≈96.53 h). Meanwhile, the highest helpfulness score received by a single review is 12,236 with the lowest being zero. The average helpfulness score is 6.42. By adding the normalized "playtime" and "voted helpfulness" values as weights to the sentiment score of each review sentence, we can obtain the weighted playability score for each perspective as follows: Functionality of $-0.1985$, Gameplay of $-0.1815$, and Usability of $-0.1983$ with the scale of $(-1, 1)$. This result shows that the overall playability of this game is slightly under mediocre. Furthermore, similar experiments with the reviews between updates show that the playability of the game is still increasing through updates, but the values are slightly negative. This phenomenon shows that experienced players and popular reviews can have obvious influence on the playability analysis result when their opinions are credited with more value. However, how to verify the influence of the players' experience and the credibility of their reviews towards the analysis result of playability shall be further investigated in future studies.

On the other hand, as shown in Table 4, the majority (61.2%) of the reviews are given before the first update of the game. Thus, the playability of Release 1.0 likely has a greater influence on the overall score than that of the rest. Therefore, it is reasonable such unevenness is also taken into account. Comparatively, for a similar situation in review-based analysis, the time sequence factor was considered by Chen et al. when evaluating the informativeness of mobile application reviews [26]. However, we are unable to conclude that the newest reviews accurately reflect the current playability of the game without further investigation on the content of such reviews compared to the older ones. A study on the changes of reviewers' opinions regarding the evolution of the target system (similar to the one in [27]) shall be conducted towards tackling such issues.

It is worth emphasizing that the proposed approach can be adapted by considering any proposed playability heuristics when such heuristic-oriented issues are sufficiently mentioned by the players. Due to the nature of heuristics being a checklist of principles [11], it is thus possible to extract players' opinions according to different heuristics via labeling training data accordingly. Although the outcome of applying different heuristics could certainly differ, it is a potentially good practice towards detecting more playability issues. Thus, a comparative study on applying this approach with different playability heuristics shall be conducted in the future work.

Furthermore, an obvious limitation of this approach is the requirement of a large number of player reviews, which is impossible before the release of the game. Heuristic evaluation of playability is an effective way to target such a situation. Comparatively, our approach aims for the continuous maintenance and evolution of games after their releases, where playability evaluation can be conveniently automated through this data mining pipeline with sufficient review data collected. The gap between experts' and end players' opinions is, to a certain extend, inevitable [52]. Hence, our approach can contribute to helping the developers better understand the needs and complaints of the players. Based on that, they can improve the games continuously and effectively.

## 6. Conclusions

In this paper, we propose a data-driven approach for analyzing the playability of video games based on the players' reviews. Focusing on the collective opinions of a large number of players, this approach provides an effective solution for understanding the overall playability of a particular video game as well as the detailed merits and defects within each pre-defined playability perspective. The results of this study show that the proposed approach can provide fair evaluation and analysis in terms of video game playability with satisfactory accuracy. Compared to the mainstream heuristic evaluation method, our approach contributes specifically to the maintenance and evolution of video games by helping game developers understand the collective needs and complaints of real players. The approach can be improved by taking into account other factors that influence the playability analysis: the playtime, voted helpfulness, player preferences, etc. The different evaluation results by selecting different playability frameworks or using different

playability heuristics shall be further investigated comparatively. Furthermore, more video game cases, especially from different genres, shall be used for verification and comparison. We shall also further investigate the credibility of game players as reviewers based on their reviewing behaviors and gaming profiles via computational methods. Such studies shall contribute to the enrichment of the playability and player behavior analysis methodologies.

**Author Contributions:** Conceptualization, X.L., Z.Z. and K.S.; data curation, X.L.; formal analysis, X.L.; investigation, X.L., Z.Z. and K.S.; methodology, X.L.; software, X.L.; supervision, Z.Z. and K.S.; validation, X.L.; visualization, X.L.; writing—original draft, X.L.; and writing—review and editing, X.L., Z.Z. and K.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the Faculty of Information Technology and Communication Sciences, Tampere University and the Ulla Tuominen Foundation.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data presented in this study are openly available in FigShare at https://doi.org/10.6084/m9.figshare.14222531.v1, accessed on 16 March 2021.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Sánchez, J.L.G.; Vela, F.L.G.; Simarro, F.M.; Padilla-Zea, N. Playability: Analysing user experience in video games. *Behav. Inf. Technol.* **2012**, *31*, 1033–1054. [CrossRef]
2. Järvinen, A.; Heliö, S.; Mäyrä, F. *Communication and Community in Digital Entertainment Services*; Prestudy Research Report; University of Tampere: Tampere, Finland, 2002.
3. Fabricatore, C.; Nussbaum, M.; Rosas, R. Playability in action videogames: A qualitative design model. *Hum.-Comput. Interact.* **2002**, *17*, 311–368. [CrossRef]
4. Desurvire, H.; Caplan, M.; Toth, J.A. Using heuristics to evaluate the playability of games. In Proceedings of the CHI'04 Extended Abstracts on Human Factors in Computing Systems, Vienna, Austria, 24–29 April 2004; pp. 1509–1512.
5. Korhonen, H.; Koivisto, E.M. Playability heuristics for mobile games. In Proceedings of the 8th Conference on Human-Computer Interaction with Mobile Devices and Services, Helsinki, Finland, 12–15 September 2006; pp. 9–16.
6. ISO/IEC. *Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE)—Measurement of Quality in Use*; Standard 25022:2016; International Organization for Standardization: Geneva, Switzerland, 2016.
7. Sánchez, J.L.G.; Simarro, F.M.; Zea, N.P.; Vela, F.L.G. *Playability as Extension of Quality in Use in Video Games*; I-USED: 2009. Available online: https://lsi2.ugr.es/juegos/articulos/iused09-jugabilidad.pdf (accessed on 16 March 2021).
8. Paavilainen, J. Defining playability of games: functionality, usability, and gameplay. In Proceedings of the 23rd International Conference on Academic Mindtrek, Tampere, Finland, 29–30 January 2020; pp. 55–64.
9. Korhonen, H.; Paavilainen, J.; Saarenpää, H. Expert review method in game evaluations: Comparison of two playability heuristic sets. In Proceedings of the 13th International MindTrek Conference: Everyday Life in the Ubiquitous Era, Tampere, Finland, 30 September–2 October 2009; pp. 74–81.
10. Pinelle, D.; Wong, N.; Stach, T. Heuristic evaluation for games: Usability principles for video game design. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Florence, Italy, 5–10 April 2008; pp. 1453–1462.
11. Nielsen, J.; Molich, R. Heuristic evaluation of user interfaces. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Paris, France, 27 April–2 May 1990; pp. 249–256.
12. Pitoura, E.; Koutrika, G.; Stefanidis, K. Fairness in Rankings and Recommenders. In Proceedings of the Advances in Database Technology—EDBT 2020, 23rd International Conference on Extending Database Technology, Copenhagen, Denmark, 30 March–2 April 2020. Available online: OpenProceedings.org (accessed on 16 March 2021).
13. Pinelle, D.; Wong, N.; Stach, T.; Gutwin, C. Usability heuristics for networked multiplayer games. In Proceedings of the ACM 2009 International Conference on Supporting Group Work, Sanibel Island, Fl, USA, 10–13 May 2009; pp. 169–178.
14. Hannula, R.; Nikkilä, A.; Stefanidis, K. GameRecs: Video Games Group Recommendations. In *ADBIS*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 513–524.
15. Gong, J.; Ye, Y.; Stefanidis, K. A Hybrid Recommender System for Steam Games. In *ISIP*; Springer: Cham, Switzerland, 2019; pp. 133–144.
16. Klancar, J.; Paulussen, K.; Stefanidis, K. FIFARecs: A Recommender System for FIFA18. In *ADBIS*; Springer: Cham, Switzerland, 2019; pp. 525–536.
17. Stefanidis, K.; Pitoura, E.; Vassiliadis, P. Managing contextual preferences. *Inf. Syst.* **2011**, *36*, 1158–1180. [CrossRef]
18. Jacobsen, N.E.; Hertzum, M.; John, B.E. The evaluator effect in usability tests. In Proceedings of the CHI 98 Conference Summary on Human Factors in Computing Systems, Los Angeles, CA, USA, 18–23 April 1998; pp. 255–256.

19. Juul, J. *Half-Real: Video Games between Real Rules and Fictional Worlds*; MIT Press: Cambridge, MA, USA, 2011.
20. CDProjekt. The Witcher 3: Wild Hunt [CD-ROM, PC, XBOX, Playstation]. 2015. Available online: https://thewitcher.com/en/witcher3/ (accessed on 16 March 2021).
21. Burnett, M.; Cook, C.; Rothermel, G. End-user software engineering. *Commun. ACM* **2004**, *47*, 53–58. [CrossRef]
22. Ko, A.J.; Abraham, R.; Beckwith, L.; Blackwell, A.; Burnett, M.; Erwig, M.; Scaffidi, C.; Lawrance, J.; Lieberman, H.; Myers, B.; et al. The state of the art in end-user software engineering. *ACM Comput. Surv. (CSUR)* **2011**, *43*, 21. [CrossRef]
23. Hearst, M.A. Untangling text data mining. In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, College Park, ML, USA, 1 June 1999; pp. 3–10.
24. Liu, B. Sentiment analysis and opinion mining. *Synth. Lect. Hum. Lang. Technol.* **2012**, *5*, 1–167. [CrossRef]
25. Fu, B.; Lin, J.; Li, L.; Faloutsos, C.; Hong, J.; Sadeh, N. Why people hate your app: Making sense of user feedback in a mobile app store. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, Chicago, IL, USA, 11–14 August 2013; pp. 1276–1284.
26. Chen, N.; Lin, J.; Hoi, S.C.; Xiao, X.; Zhang, B. AR-miner: Mining informative reviews for developers from mobile app marketplace. In Proceedings of the 36th International Conference on Software Engineering, ACM, Hyderabad, India, 1 May 2014; pp. 767–778.
27. Li, X.; Zhang, Z.; Stefanidis, K. Mobile App Evolution Analysis Based on User Reviews. In Proceedings of the 17th International Conference on Intelligent Software Methodologies, Tools, and Techniques, Naples, Italy, 15–17 September 2018; pp. 773–786.
28. Lin, D.; Bezemer, C.P.; Zou, Y.; Hassan, A.E. An empirical study of game reviews on the Steam platform. *Empir. Softw. Eng.* **2019**, *24*, 170–207. [CrossRef]
29. Nielsen, J. Usability inspection methods. In Proceedings of the Conference Companion on Human Factors in Computing Systems, Boston, MA, USA, 23–28 April 1994; pp. 413–414.
30. Nielsen, J. How to conduct a heuristic evaluation. *Nielsen Norman Group* **1995**, *1*, 1–8.
31. Korhonen, H. Comparison of playtesting and expert review methods in mobile game evaluation. In Proceedings of the 3rd International Conference on Fun and Games, Leuven, Belgium, 15–17 September 2010; pp. 18–27.
32. Malone, T.W. Heuristics for designing enjoyable user interfaces: Lessons from computer games. In Proceedings of the 1982 Conference on Human Factors in Computing Systems, Gaithersburg, Ml, USA, 15–17 March 1982; pp. 63–68.
33. Federoff, M.A. Heuristics and Usability Guidelines for the Creation and Evaluation of Fun in Video Games. Ph.D. Thesis, Indiana University, Bloomington, IN, USA, 2002.
34. Desurvire, H.; Wiberg, C. Game usability heuristics (PLAY) for evaluating and designing better games: The next iteration. In *International Conference on Online Communities and Social Computing*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 557–566.
35. Korhonen, H.; Koivisto, E.M. Playability heuristics for mobile multi-player games. In Proceedings of the 2nd International Conference on Digital Interactive Media in Entertainment and Arts, Perth, Australia, 19–21 September 2007; pp. 28–35.
36. Koeffel, C.; Hochleitner, W.; Leitner, J.; Haller, M.; Geven, A.; Tscheligi, M. Using heuristics to evaluate the overall user experience of video games and advanced interaction games. In *Evaluating User Experience in Games*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 233–256.
37. Röcker, C.; Haar, M. Exploring the usability of videogame heuristics for pervasive game development in smart home environments. In Proceedings of the Third International Workshop on Pervasive Gaming Applications–Pergames, Dublin, Ireland, 7 May 2006; pp. 199–206.
38. Tan, J.L.; Goh, D.H.L.; Ang, R.P.; Huan, V.S. Usability and playability heuristics for evaluation of an instructional game. In *E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*; Association for the Advancement of Computing in Education (AACE): Waynesville, NC, USA, 2010; pp. 363–373.
39. Khanana, K.; Law, E.L.C. Designing children's digital games on nutrition with playability heuristics. In Proceedings of the CHI'13 Extended Abstracts on Human Factors in Computing Systems, Paris, France, 27 April–2 May 2013; pp. 1071–1076.
40. Aker, Ç.; Rızvanoğlu, K.; Bostan, B. Methodological Review of Playability Heuristics. *Proc. Eurasia Graph. Istanb. Turk.* **2017**, *405*. Available online: https://www.researchgate.net/profile/Kerem-Rizvanoglu/publication/321623742_Eurasia_2017_Brave_New_Worlds_Conference_on_Virtual_and_Interactive_Realities/links/5a292400aca2727dd8872361/Eurasia-2017-Brave-New-Worlds-Conference-on-Virtual-and-Interactive-Realities.pdf (accessed on 16 March 2021).
41. Matera, M.; Costabile, M.F.; Garzotto, F.; Paolini, P. SUE inspection: An effective method for systematic usability evaluation of hypermedia. *IEEE Trans. Syst. Man, Cybern. Part A Syst. Hum.* **2002**, *32*, 93–103. [CrossRef]
42. Vasa, R.; Hoon, L.; Mouzakis, K.; Noguchi, A. A preliminary analysis of mobile app user reviews. In Proceedings of the 24th Australian Computer-Human Interaction Conference, Sydney, Australia, 20–24 November 2012; pp. 241–244.
43. Hoon, L.; Vasa, R.; Schneider, J.G.; Mouzakis, K. A preliminary analysis of vocabulary in mobile app user reviews. In Proceedings of the 24th Australian Computer-Human Interaction Conference, Sydney, Australia, 20–24 November 2012; pp. 245–248.
44. Harman, M.; Jia, Y.; Zhang, Y. App store mining and analysis: MSR for app stores. In Proceedings of the 2012 9th IEEE Working Conference on Mining Software Repositories (MSR), Zurich, Switzerland, 2–3 June 2012; pp. 108–111.
45. Vu, P.M.; Nguyen, T.T.; Pham, H.V.; Nguyen, T.T. Mining user opinions in mobile app reviews: A keyword-based approach (t). In Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), Lincoln, NE, USA, 9–13 November 2015; pp. 749–759.

46. Panichella, S.; Di Sorbo, A.; Guzman, E.; Visaggio, C.A.; Canfora, G.; Gall, H.C. How can i improve my app? Classifying user reviews for software maintenance and evolution. In Proceedings of the 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME), Bremen, Germany, 29 September–1 October 2015; pp. 281–290.
47. Gu, X.; Kim, S. what parts of your apps are loved by users? (T). In Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), Lincoln, NE, USA, 9–13 November 2015; pp. 760–770.
48. Khalid, H. On identifying user complaints of iOS apps. In Proceedings of the IEEE 2013 35th International Conference on Software Engineering (ICSE), San Francisco, CA, USA, 18–26 May 2013; pp. 1474–1476.
49. Guzman, E.; Maalej, W. How do users like this feature? A fine grained sentiment analysis of app reviews. In Proceedings of the 2014 IEEE 22nd International Requirements Engineering Conference (RE), Karlskrona, Sweden, 25–29 August 2014; pp. 153–162.
50. Li, X.; Zhang, Z.; Stefanidis, K. Sentiment-Aware analysis of mobile apps user reviews regarding particular updates. In Proceedings of the Thirteenth International Conference on Software Engineering Advances, Nice, France, 14–18 October 2018; p. 109.
51. Li, X.; Zhang, B.; Zhang, Z.; Stefanidis, K. A Sentiment-Statistical Approach for Identifying Problematic Mobile App Updates Based on User Reviews. *Information* **2020**, *11*, 152. [CrossRef]
52. Santos, T.; Lemmerich, F.; Strohmaier, M.; Helic, D. What's in a Review: Discrepancies Between Expert and Amateur Reviews of Video Games on Metacritic. *Proc. ACM Hum.-Comput. Interact.* **2019**, *3*, 1–22. [CrossRef]
53. Lu, C.; Li, X.; Nummenmaa, T.; Zhang, Z.; Peltonen, J. Patches and Player Community Perceptions: Analysis of No Man's Sky Steam Reviews. DiGRA. In Proceedings of the 2020 DiGRA International Conference: Play Everywhere, Lüneburg, Germany, 14–17 May 2020.
54. Nigam, K.; McCallum, A.K.; Thrun, S.; Mitchell, T. Text classification from labeled and unlabeled documents using EM. *Mach. Learn.* **2000**, *39*, 103–134. [CrossRef]
55. Zhang, M.L.; Zhou, Z.H. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognit.* **2007**, *40*, 2038–2048. [CrossRef]
56. Chen, W.J.; Shao, Y.H.; Li, C.N.; Deng, N.Y. MLTSVM: A novel twin support vector machine to multi-label learning. *Pattern Recognit.* **2016**, *52*, 61–74. [CrossRef]
57. Spyromitros, E.; Tsoumakas, G.; Vlahavas, I. An empirical study of lazy multilabel classification algorithms. In *Hellenic Conference on Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 401–406.
58. Gilbert, C.; Hutto, E. Vader: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). 2014; Volume 81, p. 82. Available online: http://comp.social.gatech.edu/papers/icwsm14.vader.hutto.pdf (accessed on 16 April 2020).
59. Thelwall, M.; Buckley, K.; Paltoglou, G.; Cai, D.; Kappas, A. Sentiment strength detection in short informal text. *J. Am. Soc. Inf. Sci. Technol.* **2010**, *61*, 2544–2558. [CrossRef]
60. Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent dirichlet allocation. *J. Mach. Learn. Res.* **2003**, *3*, 993–1022.
61. HelloGames. No Man's Sky [CD-ROM, PC, XBOX, Playstation]. 2016. Available online: https://www.nomanssky.com/ (accessed on 16 March 2021).
62. Syed, S.; Spruit, M. Full-Text or abstract? Examining topic coherence scores using latent dirichlet allocation. In Proceedings of the 2017 IEEE International conference on data science and advanced analytics (DSAA), Tokyo, Japan, 19–21 October 2017; pp. 165–174.
63. Greenwood-Ericksen, A.; Poorman, S.R.; Papp, R. On the validity of Metacritic in assessing game value. *Eludamos. J. Comput. Game Cult.* **2013**, *7*, 101–127.
64. Bossert, M.A. *Predicting Metacritic Film Reviews Using Linked Open Data and Semantic Technologies*; KNOW@ LOD: Portorož, Slovenia, 2015.