GOETHE UNIVERSITY
Information Systems Engineering

A thesis presented for the Master of Science Degree

App ecosystem out of balance: An empirical analysis of update interdependence between operating system and application software

---

March 31, 2020

Author:
Phillip Schneider

Supervisor:
Daniel Franzmann

Examiners:
Roland Holten
Wolfgang König

Abstract

Software updates are a critical success factor in mobile app ecosystems. Through publishing regular updates, platform providers enhance their operating systems for the benefit of both end users and third-party developers. It is also a way of attracting new customers. However, this platform evolution poses the risk of inadvertently introducing software problems, which can severely disturb the ecosystem's balance by compromising its foundational technologies. So far, little to no research has addressed this issue from a user-centered perspective. The thesis at hand draws on IS post-adoption literature to investigate the potential negative influences of operating system updates on mobile app users. The release of Apple's iOS 13 update serves as research object. Based on over half a million user reviews from the AppStore, data mining techniques are applied to study the impact of the new platform version. The results show that iOS 13 caused complications with a large number of popular apps, leading to a significant decline in user ratings and an uptrend in negative sentiment. Feature requests, functional complaints, and device compatibility are identified as the three major issue categories. These issue types are compared in terms of their quantifiable negative effect on users' continuance intention. In essence, the findings contribute to IS research on post-adoption behavior and provide guidance to ecosystem participants in dealing with update-induced platform issues.

Keywords: App ecosystem, mobile platforms, software updates, IS continuance, IS post-adoption

List of contents

List of figures

List of tables

List of abbreviations

ANOVA	Analysis of variance
API	Application programming interface
CSV	Comma-separated values
ECT	Expectation-confirmation theory
ETL	Extract-transform-load
GB	Great Britain
H	Hypothesis
HTTP	Hypertext transfer protocol
IDC	International Data Corporation
ISCM	Information systems continuance model
JSON	JavaScript object notation
KDD	Knowledge discovery in databases
LDA	Latent Dirichlet allocation
NB	Naïve Bayes
NLP	Natural language processing
NLTK	Natural language toolkit
OTA	Over the air
PC1	Principal component 1
PCA	Principal component analysis
RQ	Research question
RSS	Rich site summary
SDK	Software development kit
TF-IDF	Term frequency-inverse document frequency
URL	Uniform resource locator
US	United States
VADER	Valence aware dictionary and sentiment reasoner
XML	Extensible markup language

List of symbols

C_j	Class j
$p_j(w)$	Fraction of class j presence for word w
$w_{s,t}$	TF-IDF weighting factor for term s in document t
$x_1, x_2, \ldots, x_n$	Feature vector
x_i	Feature i
β_i	Regression coefficient for feature i
df_s	Frequency of documents containing term s
$tf_{s,t}$	Frequency of term s in document t
α	Prior of per-document topic distribution
β	Prior of per-topic word distribution
θ	Per-document topic distribution
λ	Relevance weighting parameter
ρ	Spearman's rank correlation coefficient
r	Pearson's correlation coefficient
φ	Per-topic word distribution
D	Number of documents
K	Number of classes
N	Number of words
T	Number of topics
i	Feature index
j	Class index
k	Number of folds in cross validation
n	Number of features
s	Term index
t	Document index
w	Word variable
z	Topic variable
ε	Error term

1 Introduction

Throughout the past decade, the accelerating rate of smartphone adoption has been one of the most pervasive diffusions of technology in human history. It has drastically transformed existing business models and fostered the emergence of new ones. Along with the increasing penetration of smartphones, the popularity of mobile applications has seen a remarkable growth. Mobile applications, or apps for short, are software programs designed to run on mobile operating systems, such as Apple's iOS as well as Google's Android platform. By launching the pioneering App Store marketplace in 2008, Apple was the first company to ignite a worldwide app phenomenon that has since revolutionized the industry. To this day, the App Store drives the app economy with a number of two million available applications (Apple Inc. 2019a). It counts over half a billion visitors each week and has earned more than $120 billion for developers. Digital marketplaces and operating system platforms have become cornerstones of emerging mobile app ecosystems. Similar to natural ecosystems, app ecosystems are characterized as an intricate network of various actors, including platform owners, developers, or consumers. Platform owners regularly publish software updates to enhance their mobile operating systems. However, this intentional platform evolution poses the risk of inadvertently introducing technical issues, which can severely disturb the ecosystem's balance by compromising its foundational technologies. A recent example is the release of Apple's thirteenth version of iOS. Shortly after the debut of iOS 13, hundreds of millions of iPhone users reported problems with the update, ranging from a variety of bugs to potentially dangerous security flaws. In fact, even the US Department of Defense strongly encouraged its employees not to install iOS 13 immediately, but rather to wait for the 13.1 patch (Forbes 2019). The update also drew lots of criticism from both public media and the iOS developer community. In light of the serious consequences for members of Apple's ecosystem, it is of utmost relevance to learn from the shortcomings of the iOS 13 rollout in order to avoid recurrence. With the customer base as the most valuable asset, it is especially important to analyze the concrete impact of the operating system update on app users.

A substantial body of academic literature has already explored the phenomenon of ubiquitous software updates. Although updates happen to be particularly prevalent in the mobile app sector, they are used in all areas of information technology. Developers frequently leverage them as an instrument to change the way software works by modifying its underlying source code. For instance, Apple publishes a new version of its iOS platform every year and multiple smaller patches at irregular intervals. Updates can implement novel features, fix errors, or alter the design of user interface elements. Because they play such an integral part in software management, many scholars have investigated updates in a software engineering context (Krishnan et al. 2004, p. 396; Ruhe and Saliu 2005, p. 47; Stackpole and Hanrion 2007, p. 230). Common themes are software release planning, development, or maintenance.

Despite the numerous studies about technical aspects of updates, the research situation on how individuals perceive software updates remains scarce. To address this gap in literature, some academics started to adopt a user-centered perspective. This new research stream, which gained traction over the last few years, examines behavioral dimensions and is linked to the information systems post-adoption literature. HONG et al. were among the first to inspect determinants of users' acceptance towards information systems that are modified through frequent software updates (Hong et al. 2011, p. 235). Further studies from AMIRPUR et al., FLEISCHMANN et al., and GRUPP and SCHNEIDER focused on how individuals respond to specific update types, such as feature or security updates (Amirpur et al. 2015, p. 13; Fleischmann et al. 2016, p. 83; Grupp and Schneider 2017, p. 611). Each of those studies based their research work on BHATTACHERJEE'S information systems continuance model (ISCM), a theoretical framework that explains users' intention regarding the sustained usage of information systems (Bhattacherjee 2001, p. 351). Since its explanatory power has been widely demonstrated in post-adoption literature, it is employed as theoretical lens in this thesis. Nevertheless, the existing research work mainly deals with direct update effects confined to one software system. To the author's best knowledge, not a single study has yet addressed the complex interdependency between mobile platform updates and application software from a user-centered perspective. This research gap is astonishing, as virtually all consumer applications rely on a functioning operating system layer. If the latter is impaired, as in the case of the iOS 13 update, end users can be confronted with countless software failures. Therefore, the present thesis seeks to answer the following research question:

RQ: What are the influences of mobile operating system updates on app users?

In accordance with the research question above, the goal of this thesis is to empirically analyze the adverse impact of the iOS platform update on end users in Apple's ecosystem. The study results are expected to contribute to the theoretical topic of software updates by providing a better understanding of how platform users perceive update-induced issues and how their attitude changes in reaction to the introduced problems. In this way, the acquired findings about the complex relationship between platform evolution and post-adoption behavior can constitute as an important input for information systems literature. They can also serve as a foundation for follow-up studies on this topic. Furthermore, by shedding light on negative effects of software updates within the mobile ecosystem, in particular the manifold technical complications reported by end users, the thesis offers valuable insights for app developers and platform owners alike. This practical knowledge can help actors of mobile app ecosystems to better cope with such problems or to avoid them in the first place. Consequently, actors can use the recommended actions derived from the research results to take preventive measures to collectively ensure the ecosystem's stability.

To meet the objectives of the thesis at hand, an exploratory research approach is employed. In consideration of the ISCM as theoretical lens, several hypotheses are deduced that attempt to answer the underlying research question. The hypotheses are subject to empirical testing. In this regard, vast amounts of publicly available customer reviews are collected from Apple's App Store through web scraping. Following the conventional data mining procedure, the scraped review data first undergo some preparation, aimed to transforming them into an adequate format for the ensuing data processing phase. Then, data mining techniques like sentiment detection, text classification, and topic modeling assist in the extraction of relevant patterns and useful information out of app ratings, review content, and textual sentiment. Together with statistical analyses, the empirical evidence from the mined feedback data are involved in the verification of the formulated research hypotheses. In a last step, both theoretical and practical implications are inferred from the obtained results.

The structure of this thesis is composed of seven chapters. The introductory section of chapter 1 was presented above. Chapter 2 provides the theoretical background and a literature review. The key concepts of mobile app ecosystems and software updates are defined, followed by a detailed explanation of the ISCM. For each of those main topics, related studies are discussed. Chapter 3 builds upon the theoretical foundations of the previous chapter and involves the development of hypotheses. In chapter 4, a description of the research methodology and design is included. Because the study design reflects the four steps of the traditional data mining process, such as data collection, preparation, processing, and evaluation, a separate subchapter is devoted to each step. Chapter 5 concerns the presentation of the analysis results with various diagrams and tables. Chapter 6 further elaborates upon the empirical results in an extensive discussion of the findings with respect to the thesis statement. It also contains the examination of the postulated hypotheses. In the last section, chapter 7, the key findings and important contributions to theory and practice are summarized, whereby potential limitations of the results are set out. Finally, the thesis concludes with an outlook on possible future research objectives.

2 Theoretical background and literature review

In order to analyze update interdependence within mobile app ecosystems, it is indispensable to understand corresponding key terms, concepts, together with their definitions, and theoretical frameworks. Hence, this chapter first gives an overview of the development and inner mechanisms of mobile app ecosystems. Apple's ecosystem, as representative research subject, is described in more detail. Second, an explication of software updates in general and a three-class update typology are presented. The last section introduces the ISCM and its psychological constructs. In addition, relevant scholarly literature is discussed for each concept.

2.1 Mobile app ecosystems

The ecosystem concept is fundamental to several scientific areas, including business and technology. However, the term originated in the science of ecology. It was coined by the English botanist TANSLEY in 1935 to designate a level of organization which integrates both living and non-living components of communities and their environments into a functional unit (Tansley 1935, pp. 299–300). All ecosystems function via the interactions between their constituents. To achieve stability, an ecosystem must reach a state of dynamic equilibrium, whereby a self-regulating balance exists between the components.

Owing to its emphasis on interconnectedness, cooperation, and competition, the ecosystem concept provides a useful analogy for understanding complex affairs in business environments. The idea of applying ecological metaphors to the business world was first put forward by MOORE who developed the theory of business ecosystems (Moore 1993, p. 75). In his Harvard Business Review paper from 1993, he argues that successful companies are those that are evolving quickly, since the only sustainable competitive advantage comes from greater innovative strength. Additionally, he proposes that innovative businesses cannot evolve in a vacuum but rather coevolve capabilities and align themselves with other members of the ecosystem. The latter include competitors, suppliers, manufacturers, and other stakeholders. Together they form an economic community producing products as well as services for consumers (Moore 1996, p. 16). By way of example, MOORE cites the technology company Apple as a successful ecosystem leader crossing several industries like consumer electronics, software or online services. Through its cultivated community of partners, suppliers, and customers, Apple's leadership is valued by all ecosystem members in respect of investing towards a shared future (Moore 1993, p. 76). Along the same lines, IANSITI and LEVIEN elaborate on the business ecosystem concept and list two crucial factors for ecosystem wellbeing (Iansiti and Levien 2004, pp. 43–47). First, business ecosystems consist of a diverse network of contributors relying on each other's mutual performance. While each member is specialized in a certain field, it is the collective effort of the community that

creates value. Second, ecosystems require so called keystone organizations whose role is to ensure good ecosystem health by managing important resources and shaping the network structure. For example, keystones establish platforms such as marketplaces, services, or technologies which help community members to enhance their own performance. Consequently, a keystone players' strategy is of central importance for all ecosystem stakeholders.

As the software industry has become increasingly interdependent, the shift of focus from the individual software organization towards an ecosystem of interrelated organizations with shared platforms is evident. In this context, platforms represent foundation technologies that link actors within an ecosystem and reduce frictions for interactions to take place. Each platform holds a set of access points or interfaces made available for ecosystem members (Iansiti and Levien 2004, p. 148; Tiwana et al. 2010, p. 675). The emergence of software ecosystems, which are subsets of business ecosystems, was first addressed in 2003 by MESSERSCHMITT and SZYPERSKI, but it took until 2009 before precise definitions in literature were formed (Messerschmitt and Szyperski 2003, p. 3). To cite a widely accepted definition, the authors JANSEN et al. delineate a software ecosystem as "a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. These relationships are frequently underpinned by a common technological platform or market and operate through the exchange of information, resources and artifacts" (Jansen et al. 2009, p. 35). Similarly, BOSCH declares a software ecosystem as consisting of the set of software solutions that aid the activities of the members in the associated ecosystem as well as the central organizations providing these solutions (Bosch 2009, p. 112). He also introduces a software ecosystem taxonomy. His two-dimensional taxonomy distinguishes between the level of abstraction and the dominant computing platform.[1] For this thesis, the category of operating system-centric mobile software ecosystems is specifically relevant.

Mobile app ecosystems can be referred as subtypes of software ecosystems related to mobile applications. The ecosystem perspective is well transferable to the mobile sector (Basole 2009, p. 147; Peppard and Rylander 2006, pp. 130–131). Driven by technological evolution, growing consumer demand, and the formation of strategic alliances, an intricate network of various actors has formed around this profitable market. Undoubtedly, these dynamics shaped a competitive market structure where firms must cope simultaneously with innovation pressure, changing customer expectations, and regulatory influences in order to survive. Numerous studies about the mobile industry have applied the ecosystem concept. Some have outlined the role of specific actors, such as device manufacturers or app developers (Dittrich and Duysters 2007, p. 510; Qiu et al. 2017, p. 225). Others examined the products and services like mobile content or collaborative service systems (Eaton et al. 2015, p. 217; Peppard and Rylander 2006, p. 130). Taken as a whole, these studies suggest that the mobile app

[1] For a detailed overview of the software ecosystem taxonomy, refer to Appendix B.1.

ecosystem consists of three major players: the customer, the ecosystem orchestrator, and the complementors (Manikas and Hansen 2013, pp. 1300–1302). The first of these is the purchasing end customer, whereas the ecosystem orchestrator is the keystone organization keeping the whole ecosystem intact through regulated ownership of the platform technologies as, for instance, Apple or Google. Complementors offer products or services that complement the product portfolio of the ecosystem orchestrator by adding value to a common customer base. These include infrastructure providers, device fabricators, and app developers, the latter being the primary focus of this study. For the sake of clarity, Figure 1 gives a conceptual overview of the mobile app ecosystem.

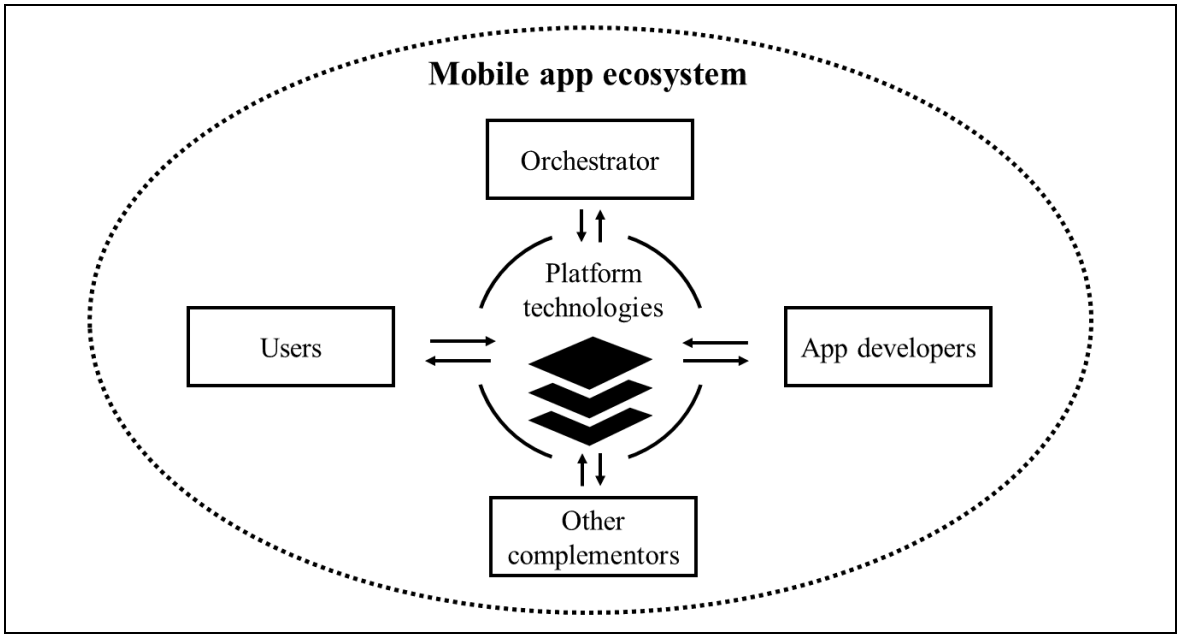


Figure 1: Conceptual model of mobile app ecosystems

As can be seen in the conceptual model above, the ecosystem, represented by the dotted ellipse form, evolves around the already mentioned actors and foundational platform technologies through which they interact with each other. One essential platform in app ecosystems is the operating system governed by the orchestrator. Operating system providers offer application programming interfaces (APIs) and development tools to simplify the adoption of the operating system by developers (Bosch 2009, p. 113). Then, app developers build applications on top of it in order to extend its functionality for users. Although the operating system offers generic functionality, it constantly needs to advance via updates, since it must maintain both a high number of end users and attractiveness for developers due to the ever-present competition with other mobile ecosystems. However, this software evolution process can easily become a major source of inefficiency. Without careful software quality management, new updates may imply compatibility issues disrupting the susceptible balance of the ecosystem. Herein lies a major risk for all involved actors.

Apart from operating systems, digital app distribution channels, also referred to as app stores, are a crucial component of the mobile ecosystem. Taking Apple's ecosystem as object of investigation, both platform technologies are discussed in the following chapter.

2.1.1 App Store and iOS platform

The continuing expansion of mobile app ecosystems led to the rise of two predominant players. The two technology companies Apple and Google established their dominance through their platforms iOS and Android, respectively. Together, their operating systems had more than 99% of the market share in 2018, leaving an insignificant share for other platforms (IDC Inc. 2020). Because this thesis concentrates on Apple's ecosystem as research subject, the subsequent paragraphs elaborate on Apple's iOS platform and its affiliated App Store.

Marking a major milestone in mobile history, Apple unveiled the initial iOS version for the original iPhone in June 2007. As first commercial mobile operating system, it comprised a software platform on top of which other application programs could run on mobile devices. It brought many revolutionary concepts, including badge notifications, slide to unlock, or multi-touch. At first, iOS did not support native third-party applications. Only a dozen preinstalled apps were usable, for example, Calendar, Notes, and Weather (Helal et al. 2012, p. 9). Apple executives argued that developers could create web-based applications that would behave like the existent native iPhone apps, which were written in iOS specific programming languages. In the light of displeased developers and the announced Android platform, Apple released a software development kit (SDK) for iOS in 2008. The SDK allowed developers to access a development environment, a user interface configurator, and a simulation tool for application testing (Ghazawneh and Henfridsson 2013, p. 182). Later in the same year, Apple also launched the App Store where developers could publish their applications to users of iOS devices. It provided a joint platform through which both developers and consumers could connect. This new digital marketplace led to a massive change in the mobile device sector. It should soon become one of the most distinguishing elements of the mobile app ecosystem.

By launching the App Store, Apple followed the idea of creating a curated distribution channel integrated in every single iPhone, so developers would be able to reach every user. Apple took on the role of curator enforcing rigid quality controls on the submitted apps. The application review policies entailed checking an applications compliance with regulations and guidelines. For instance, submissions with banned content types or malicious software were persistently rejected. Many third-party developers criticized the application review process as being too restrictive. In compensation, the SDK got radically updated with a multitude of new features and APIs as an attempt to better address the heterogenous developers' needs (Ghazawneh and Henfridsson 2013, pp. 183–185). As both large and small app publishing

firms entered this dynamic new market, the growing developer base resulted in a greater variety of apps available to customers. This branched out the iPhone scope into new application areas (Garg and Telang 2013, p. 1254). For instance, almost five years after its opening along with an inventory of only 500 apps in 2008, the App Store offered more than 850,000 mobile applications to users in 155 countries around the world. Apple customers could already choose from a wide range of 23 categories, including Business, Games, Health & Fitness, or Travel. The download rate amounted to 800 apps per second converting to two billion apps per month at that time (Apple Inc. 2013).

In short, Apple's described App Store strategy proved to be exceptionally successful. It was a balance act between regulation-based securing and diversity securing. On the one hand, Apple exercised control over the platform as regulating orchestrator. Strict App Store policies were imposed to make the platform a safe and trusted marketplace with high quality apps for users. The updated SDK with more APIs, on the other hand, were actions to resource the new device capabilities and to better support the developer community (Ghazawneh and Henfridsson 2013, p. 187). Therefore, app stores can be seen as a catalyst facilitating interactions amongst members of the same platform. Reducing transaction costs, building audiences, or matchmaking are just a few exemplary aspects. Through the creation and sharing of value, they enable the corresponding owners to manifest their keystone advantage within the mobile app ecosystem (Iansiti and Levien 2004, p. 44).

Without doubt, the App Store success contributed substantially to the fact that more and more customers bought iPhones and thus joined Apple's ecosystem. Due to the popularity of iPhones, Apple decided to use the iPhone operating system for its remaining portfolio. Today, iOS runs on iPads, iPods, and the Apple TV box. However, each device runs with a specific iOS customization (Helal et al. 2012, p. 9). The maintenance of the iOS platform is subject to high standards. Apple releases updates on a regular basis throughout the year, sometimes to add new features and sometimes to fix bugs. In general, iOS updates can be categorized into three types with regard to their three-part version number. First, a major update happens once a year. It usually includes profound system changes like, for example, the alteration of the processor architecture between iOS 10 and iOS 11. Second, a minor update, such as from iOS 11.1 to iOS 11.2, can occur multiple times per year and extends functionalities on a larger scale. Last, patch level updates, as from iOS 11.2.1 to iOS 11.2.2, occur at irregular intervals. They reflect bug fixes or urgent security patches (Apple Inc. 2019b). The most recent major iOS update was released on 19 September 2019. Being the thirteenth version, iOS 13 brought, for instance, a dark mode color scheme, improvements for Apple's digital assistant, or privacy enhancing upgrades. Despite the dozens of new fea-

tures, the iOS update was accompanied with many issues forcing Apple to release counteracting updates at a higher-than-usual frequency. The impact of these update-induced platform issues on the sensible ecosystem balance are examined in detail in later chapters.

2.1.2 Studies related to mobile app ecosystems

Software ecosystems and their inner components have been widely investigated in previous work. While some researches discuss mechanisms of ecosystems in general, others concentrate on certain app ecosystems. Below, some relevant findings are summarized.

An empirical study of CECCAGNOLI et al. addressed the question whether participation in a platform-based ecosystem improves the business performance of small software vendors (Ceccagnoli et al. 2012, p. 263). After analyzing partnering activities and performance indicators of a sample of 1,210 software vendors, the researchers found that joining a platform ecosystem was associated with gains in operational performance and a greater likelihood of obtaining an initial public offering. Hence, they empirically demonstrated that, on average, varied benefits could accrue to firms participating in a platform-based ecosystem.

Other research work focused specifically on the interaction between ecosystem members. By exploring ecosystem interrelations between app developers and users with time-series analysis, SONG et al. investigated how cross-side network effects are influenced by the platform orchestrator's governance rules (Song et al. 2018, p. 121). Their results indicated that while long application review time enhanced the quality of applications, the delay also negatively impacted the user side, which though might be counteracted by the former effect. Additionally, frequent platform updates compelled developers to commit more efforts to updating existing apps rather than developing new ones.

In a longitudinal measurement study, WANG et al. analyzed the evolution of mobile ecosystems over more than three years (Wang et al. 2019, p. 1988). Based on 5.3 million app records gathered from three Google Play snapshots, the scholars observed favorable progress in respect of app popularity, user ratings, or privacy policies, although there remained numerous unsolved issues, including update problems, malicious apps, and improper promotion actions. By shedding a light on these dynamics, the study aimed to assist developers in making better decision and give insights for app market owners to detect misbehaviors.

GENC-NAYEBI and ABRAN carried out a methodical literature review about opinion mining from mobile app store user comments (Genc-Nayebi and Abran 2017, p. 207). The objectives were to identify proposed solutions for opinion mining as well as existing challenges and problems in the research field. In total, the scholars selected a total of 24 relevant studies

for their literature review. Even though they pointed out the problem of spam or fake reviews, they concluded that customer reviews contained valuable information about user expectations, which both developers and app store regulators could leverage to better understand their customer base.

EATON et al. conducted an embedded case study of Apple's iOS ecosystem with regard to the software tools and regulations that serve as the interface between platform owner and application developers (Eaton et al. 2015, pp. 217–218). They wanted to understand how these so-called boundary resources come to be and evolve over time. Their in-depth analysis of 4,664 blog articles with 30 boundary resources suggested that part of Apple's success could be subscribed to a distributed, heterogeneous pool of actors who collectively tune boundary resources rather than Apple's sole control over the iOS ecosystem. Since the authors pointed out the cocreative nature of boundary resources, they contributed to existing mobile ecosystem research literature.

Another study focusing on Apple's ecosystem was performed by GARG and TELANG (Garg and Telang 2013, p. 1253). Taking the App Store as subject of interest, they presented an innovative approach to infer app rank-demand relationships from publicly available data. According to their findings, an iPhone app ranked first got 150 times more downloads than the app ranked at place 200. Also, the app ranked at 200 on the top grossing list earned 95 times less revenue compared to the app ranked first. The constructed model provided a new way to estimate the products sales from app rankings.

Overall, mobile app ecosystems have been examined from various angles and in the light of diverse contexts. In spite of this, the search for studies that combine the two aspects of evolving platform technologies and user behavior consequent to updates was unsuccessful. The thesis at hand tries to overcome this research gap.

2.2 Software updates

Updates play an integral part in the software maintenance process, as they enclose addition, deletion, or refining activities. Software updates can be regarded as modifications of the way software works by changing the underlying base software. Contrary to stand-alone programs, they directly integrate into deployed software that is already in operation. The update procedure must therefore document each incremental change, which is usually expressed in a different version number, and control for potentially introduced errors (Stackpole and Hanrion 2007, pp. 165–166). In practice, update requirements have been growing at a remarkable pace. All kinds of software are reliant on subsequent alterations, ranging from applications, drivers to operating and information systems. Within the domain of mobile app ecosystems, software updates are particularly prevalent. For example, users of either Android or iOS

phones receive application and operating system updates on a regular basis. Those updates are distributed automatically over the air (OTA), while in the past updates had to be manually installed from data storage media. OTA updating for mobile devices lets users download and install actualized software versions over a wireless Internet connection like mobile broadband (Herle and Fan 2010, pp. 1–2). The mechanism of OTA provisioning is indispensable for every mobile app ecosystem.

In the sphere of information technology, the pervasive usage of updates is accompanied by a large body of research. Because of varying preferences of terminology in literature, it is advisable to differentiate some key concepts. The term software patch is used interchangeably with software update. Patches, or updates, are in general free of charge and provide security improvements, bug fixes, and many other services (Kushwaha et al. 2012, p. 20). Unlike updates, software upgrades fall into the category of chargeable software services. Once users have paid for the upgrade, they get an extended set of new features that are unavailable in the not upgraded standard version. This enlarged scope sometimes implies a full re-installation of the product replacing the older version.

Nonetheless, the benefits of updating come with a trade-off. Downloading patches consumes device resources together with network bandwidth, what can cause problems in case of larger update files. Furthermore, modifications to the installed base software must be handled with caution (Kushwaha et al. 2012, p. 21). The same applies in view of dependencies between software packages. Otherwise, not only stability or performance can be impacted, but even the compatibility and basic operability. In the worst case, update-induced errors can lead to a total system failure. Thus, software developers are bound to face the dilemma of quickly releasing and carefully testing recent software versions at the same time (Stackpole and Hanrion 2007, pp. 166–167).

2.2.1 Overview of different update types

Research on software updates mainly distinguishes between three distinct types, namely feature, non-feature, and security updates (Franzmann et al. 2019, p. 5). For this reason, a detailed explanation of this three-class typology is necessary.

As the name indicates, feature updates introduce new features to the running version by changing its core functionality. They are used by developers to add, remove, or modify the current feature set of the underlying base software. Especially within the mobile ecosystem, feature updates have been increasingly leveraged by developers over the last years (Amirpur et al. 2015, p. 2). To cite a popular example, Apple’s mobile operating system gets a major update every year. Its most recent version iOS 13 added, among other features, a dark mode color scheme, a swipe keyboard, and novel photo editing tools (Apple Inc. 2019b). From the

user's viewpoint, feature updates are either proclaimed through notifications or recognized due to visible changes. Although customers need to adjust the way how they interact with the updated version, feature updates normally correlate with higher perceived satisfaction and practicality (Fleischmann et al. 2016, p. 83).

Non-feature updates represent the second class in the update typology. Without affecting the software's core feature set, they rather concentrate on correcting known errors or stability deficiencies. Such actualizations include bug fixes, more urgent hot fixes, as well as performance patches. Highlighting the difference relative to feature updates, AMIRPUR et al. point out that non-feature updates "[…] often do not directly affect the user's interaction with the software and are typically not even visible to the user […]" (Amirpur et al. 2015, p. 1). To continue the example from above, iOS 13 received numerous patches for bug fixes and improvements after its initial release due to a high incidence of errors (Apple Inc. 2019b). Therefore, non-feature updates occupy a central position in software maintenance.

Security updates represent a subclass of non-feature updates, since they neither add functionalities nor contain apparent modifications. Their purpose lies in guaranteeing protection against malicious code and other threats by closing security breaches (Grupp and Schneider 2017, p. 613). Since both organizations and individuals heavily rely on the integrity of information systems, security patches are widely studied in the literature (Cavusoglu et al. 2008, p. 657; Ng et al. 2009, p. 815). In the example case of iOS 13, Apple detected a major security flaw giving third-party keyboards full access to external services without the user's knowledge. Affected keyboard apps could record and upload sensitive data. As a consequence, Apple released a global warning message and iOS version 13.1.1 to patch this vulnerability (Apple Inc. 2019c).

2.2.2 Studies related to software updates

Academic literature has comprehensively dealt with software updates. This subchapter aims to give a concise overview of related work. To build a general understanding about the innate characteristics of updates, studies from the two main research streams regarding technical and user-centered literature are presented. In the software engineering literature, scholars focus on the technical aspects about patches. Here, the course of action from the developer standpoint forms the context. Common research topics are software release management, development, or maintenance.

Release management plays a central role in software engineering projects. It refers to decisions about software release cycles under technical, budget, and risk constraints. The key tasks are planning, testing, and deploying updates to software systems while minimizing business disruptions (Stackpole and Hanrion 2007, p. 230). A study from RUHE and SALIU

investigated best practices in the release planning process (Ruhe and Saliu 2005, p. 47). They proposed a hybrid planning approach combining benefits of human experience together with computational intelligence. Introducing more formalism for transparency, setting release planning goals with several impacting criteria, or the pro-active evaluation of possible release strategies were among their recommendations for organizations. Failing this, the researches listed unsatisfied customers, missed delivery deadlines and poor quality standards as implications of inadequate release management.

Software development and maintenance differ from release planning, since they take up later stages within the software life cycle. The purpose of maintenance activities is to modify and perfect software systems after delivery. Rolling out patches is one of the most common maintenance tasks (Stackpole and Hanrion 2007, pp. 166–167). In a paper published in Information Systems Research, KRISHNAN et al. drew attention to the fact that software maintenance accounts for a substantial amount of overall life cycle costs (Krishnan et al. 2004, p. 396). As a measure for cost reduction, the researchers came up with a stochastic decision model for maintenance of information systems. This dynamic framework could compute the optimal state for a major system update, based on the uncertainties in the user environment and product performance (Krishnan et al. 2004, p. 410).

Apart from the technical literature on software updates, a second research stream gained traction over the last few years. Rather than focusing on the developer perspective, recent studies put efforts into understanding the user's perception concerning updates, in which the context revolves around post-adoption behavior. A study conducted by HONG et al. was among the first to examine users' intention to continue using software systems that received feature updates (Hong et al. 2011, p. 266). After the analysis of data from 477 users, the scholars discovered that the user's level of comfort with ongoing changes represents a notable driver of information system acceptance.

Having a similar research interest, FLEISCHMANN et al. explored the relationship between updates and information system continuance intention by carrying out a controlled laboratory experiment (Fleischmann et al. 2016, p. 83). Based on their experimental results, the scholars found a positive effect of feature updates on users' willingness to continue using information systems. This effect further intensified with more frequent feature updates. In addition, the study unveiled that this effect operates through positive disconfirmation, leading to higher levels in perceived usefulness and satisfaction. Further evidence in support of these findings was found in another study from AMIRPUR et al. (Amirpur et al. 2015, p. 13). Consequently, FLEISCHMANN et al. advised software vendors to leverage the gained insights in practice. Even though the authors did not find the same effect for non-feature updates, a

following study from FRANZMANN et al. demonstrated that non-feature updates, such as bug fixes, can have a positive effect on users' impression as well (Franzmann et al. 2019, p. 3).

In accordance with the study results above, GRUPP and SCHNEIDER empirically confirmed that feature patches elicit a positive effect on users' continuance intention (Grupp and Schneider 2017, p. 611). Moreover, based on their online experiment with 282 participants, they could even prove a positive impact of security updates on users' continuance intention, as long as users got a notification after successful implementation. Therefore, they suggested practitioners should always inform users about new patch releases.

Summarizing the literature review, despite the vast dissemination of software updates, existing studies are mostly oriented towards a technical background, whereas literature on how individuals perceive updates is still scarce. Lately, some academics started to address this gap in literature by adopting a user-centered approach. For instance, AMIRPUR et al., FLEISCHMANN et al., FRANZMANN et al., and GRUPP and SCHNEIDER explicitly investigated the influence of different update types on users within the post-adoption phase (Amirpur et al. 2015, p. 13; Fleischmann et al. 2016, p. 83; Franzmann et al. 2019, p. 1; Grupp and Schneider 2017, p. 611). They all based their work on the information systems continuance model, which was also employed as a research lens for the present thesis. The next section elaborates on the properties of this model.

2.3 Information systems continuance model

The information systems continuance model (ISCM) is an influential theory in post-adoption literature. This research stream looks into situations where users are confronted with the decision to either continue or stop using an information system after they have initially accepted it (Hong et al. 2011, p. 238). To study user intentions regarding sustained usage of software systems, BHATTACHERJEE was the first to adapt the expectation-confirmation theory (ECT), which serves as theoretical basis for the ISCM, for an information systems context (Bhattacherjee 2001, p. 351).

Providing a cognitive framework to study customer satisfaction and post-purchase behavior, ECT has been widely used in consumer psychology literature and marketing in general (Anderson and Sullivan 1993, p. 125; Oliver 1993, p. 418). In the year 1977, a foundational experiment from OLIVER studied the two constructs expectation and disconfirmation on perceived product performance (Oliver 1977, p. 480). His survey with 243 respondents, who reacted to a recently introduced product, demonstrated that disconfirmation was not correlated with expectations measured before revealing the product, thereby allowing room for an additive interpretation. Following up on these findings, OLIVER wrote a second paper in 1980 proposing a research framework in which satisfaction is expressed as a function of

expectation and disconfirmation (Oliver 1980, p. 460). Satisfaction, in turn, was determined as an influencing factor of the repurchase intention. Briefly explained, consumers possess certain expectations about a product or a service before buying. Once they have collected experiences with it, they compare the perceived performance with preceding expectations. If the former exceeds the latter, the disconfirmation is positive implying a positive effect on satisfaction. Conversely, if consumers' expectations are negatively disconfirmed due to underperformance, satisfaction decreases. Lastly, the repurchase or reuse intention depends on the level of satisfaction. To date, the explanatory power of ECT was verified across various experiments (Liao et al. 2017, p. 651; Oliver 1993, p. 418; Patterson 1997, p. 4).

Based on the ECT framework and prior information systems literature, BHATTACHERJEE conceptualized the ISCM in an attempt to better understand users' intention to continue using information systems (Bhattacherjee 2001, pp. 351–356). His model, which is illustrated in Figure 2, theorizes the difference between acceptance and continued usage by examining personal attitudes and beliefs. Hence, subject of interest in the ISCM is the intention of sustained continuance instead of the repurchase inclination. As displayed below, the ISCM encompasses four primary constructs: disconfirmation, satisfaction, perceived usefulness, and continuance intention, the latter being self-explanatory. Disconfirmation stands for the discrepancy in users' observation between product expectation and actual performance. Satisfaction is the affective evaluation of prior usage. Perceived usefulness, an adapted construct of the technology acceptance model, symbolizes expected benefits of post-adoption usage (Davis et al. 1989, p. 982). All model constructs are post-acceptance variables, since disconfirmation and satisfaction already capture pre-acceptance effects.

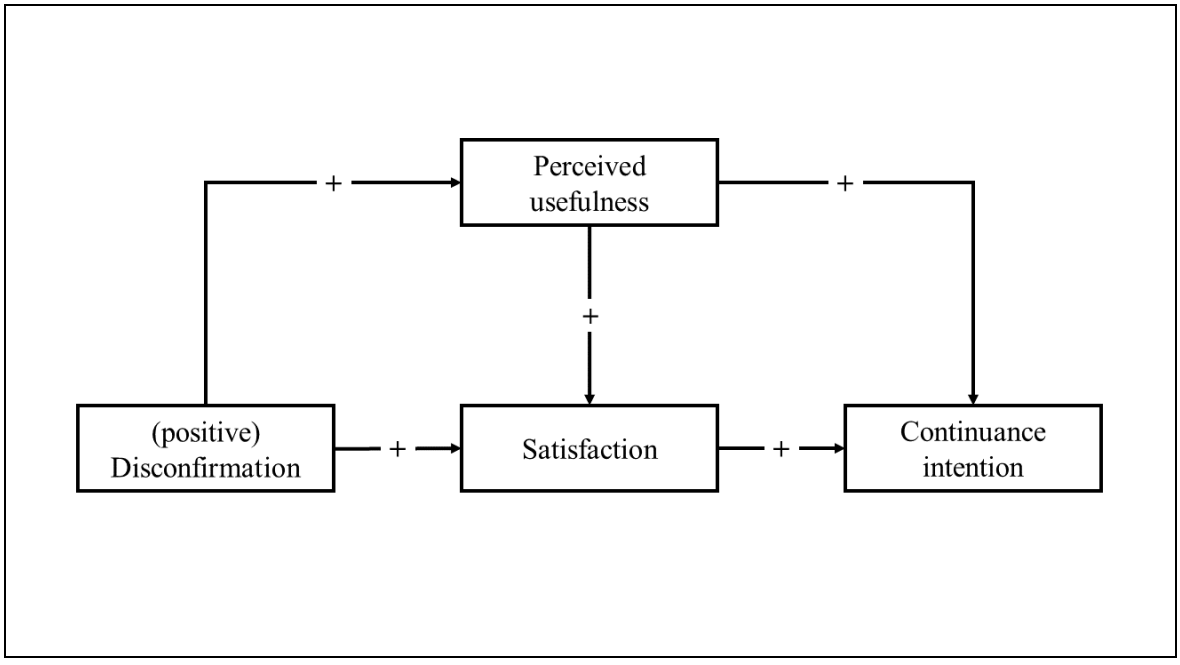


Figure 2: Information systems continuance model (Bhattacherjee 2001, p. 356)

The underlying reasoning of the ISCM functions in a similar way as the ECT framework. In his model, BHATTACHERJEE postulates that the predicted continuance intention depends primarily on satisfaction and secondary on perceived usefulness (Bhattacherjee 2001, pp. 364–367). This is because perceived usefulness is more central to acceptance than continuance intention. User satisfaction, for its part, is determined by disconfirmation and perceived usefulness, while disconfirmation has a relatively stronger impact. Perceived usefulness is solely influenced by disconfirmation, whereby positive disconfirmation increases the level of perceived usefulness. Ultimately, the construct of disconfirmation can be interpreted in accordance with the ECT. If the judged performance of an information system surpasses users' original expectations, they are positively disconfirmed which, as a direct consequence, elevates their satisfaction and perceived usefulness. The reverse causes negative disconfirmation, dissatisfaction and thus indirectly the intention of discontinuance.

Since its introduction in 2001, the ISCM gained prominence in information systems literature. It has been widely applied and expanded by scholars to study the phenomenon of post-adoption behavior across a variety of settings (Benlian et al. 2011, p. 85; Chang and Zhu 2012, p. 995; Larsen et al. 2009, p. 778; Limayem et al. 2007, p. 705). For instance, HONG et al. extended the ISCM with components from the technology acceptance model to examine the continuance intention of mobile Internet usage (Hong et al. 2006, p. 1819). In addition, BHATTACHERJEE and PREMKUMAR proposed an augmented ISCM with a pre-usage and usage stage to understand shifts in users' attitudes and beliefs over time (Bhattacherjee and Premkumar 2004, p. 229). Thinking one step further, along with changes in users' perception, it is rational to presume that the information system itself is exposed to temporal changes, for example, through software patches. Both of these dynamic changes influence users' continuance intention (Fleischmann et al. 2016, pp. 88–89; Ortiz de Guinea and Webster 2013, p. 1165). For the aspects cited above, the ISCM was selected as solid theoretical lens for exploring post-adoption behavior regarding updates within mobile app ecosystems in this thesis.

3 Development of hypotheses

The following chapter is dedicated to formulating educated guesses in connection to the research question. It contains derivations of hypotheses about the influence of evolution-induced platform issues on mobile app users. Each deduced hypothesis rests upon the theoretical foundations from chapter 2. In the center of investigation stands the operating system update iOS 13 and its adverse impact on users in Apple's mobile ecosystem.

3.1 Adverse effects of update-induced issues on continuance intention

As detailed in section 2.1, mobile app ecosystems are characterized by a fragile balance between actors and platform technologies through which they interact. This applies in particular to the continuous evolution of mobile operating systems. Software patches provide incremental platform improvements that aim to strengthen the ecosystem as a whole (Bosch 2009, p. 113). Yet, herein lies the risk of destabilizing the ecosystem by accidentally introducing detrimental software flaws, as it was the case with Apple's current mobile operating system update. The patch release documentation of iOS 13 gives a rough idea about the wide scope of compatibility, performance, and security problems (Apple Inc. 2019b). Chapter 1 has highlighted the consequent severe negative feedback from both public media as well as app users. Given the interdependent nature of mobile ecosystems, the update-induced issues assumingly impacted a huge number of app users. Related information systems literature has already demonstrated that certain software updates might diminish customers' continuance intention, alienate existing ones, or even increase their inclination of switching to a competitor (Fleischmann et al. 2015, p. 17; Foerderer and Heinzl 2017, p. 1). In line with the ISCM, the psychological mechanism behind these user responses operates through negative disconfirmation regarding the updated software. This rationale can be transferred to the indirect adverse effects of Apple's iOS update on mobile applications and their users. Following this argumentation chain, it is hypothesized that:

H1: The release of the new operating system update introduced issues that caused a significant decline in app users' continuance intention.

3.2 Manifestation of distinct update issue types in user reviews

One distinguishing element of the mobile ecosystem is its digital app distribution channel, just like the App Store from Apple. In this online marketplace, developers publish applications for users to download and rate. More specifically, App Store customers have the option to rate content on a scale of 1 to 5 stars, with 5 being the best. Customers are allowed to complement their reviews with a short text about their personal evaluation. Recent work shed light on the valuable insights contained in these user reviews, which both app store

owners and developers can leverage (Chen et al. 2014, p. 767; Genc-Nayebi and Abran 2017, p. 207; Noei et al. 2019, p. 1; Panichella et al. 2015, p. 281). Such customer feedback can address diverse topics, for example, feature requests, functional complaints, detected bugs, privacy concerns, and other problem cases. Considering the many issues related to iOS 13, it seems sensible to assume that a myriad of app users wrote reviews about the complications they faced. Hence, in accordance with previous work on app review mining, major problem categories reflected in user feedback should be theoretically identifiable with adequate text mining methods, which results in the supposition that:

H2: Distinct update issue types are manifested in user reviews.

3.3 Magnitude of influence of issue types on continuance intention

Based on the statements above, the iOS 13 update was accompanied by a considerable number of distinguishable app issues, leaving frustrated and dissatisfied customers. Drawing from information systems post-adoption literature, users' perception of modified software varies depending on the degree of disconfirmation (Fleischmann et al. 2016; Hong et al. 2011, p. 238; Limayem et al. 2007, p. 733). While updates are usually not anticipated by customers, unexpected update-induced errors during usage exert negative disconfirmation, which eventually leads to declining usage rates or even complete discontinuation. It appears plausible that the more critical an experienced error is, the more negative will be the user response. This logic can be applied to the context of iOS 13. As indicated in Apple's iOS patch documentation, the criticality of errors is variable, ranging from minor performance lags to major software failures and security breaches (Apple Inc. 2019b). It is therefore likely that some specific issue categories are correlated with a lower continuance intention than other categories. Hence, grounded on the outlined reasons, it is expected that:

H3: The influence of different issue types on app users' continuance intention varies in terms of magnitude.

4 Research methodology

This chapter concentrates on the research methodology undertaken in the present thesis. It begins with a discussion of the general study design, followed by a description of the data collection process. Then, the executed data preparation steps are explained one by one. The subsequent section introduces the applied data mining methods. Lastly, the analytical evaluation of the extracted data patterns is discussed.

4.1 Study design

Against the background of the research question, the study design articulates the overall strategy that combines multiple methods and techniques of the study in a coherent way, thereby ensuring the scientific verification of proposed hypotheses. Given the plethora of design options, it is fundamental to select a design fitting the study objectives in the best possible manner in consideration of all limitations. Research methods are broadly categorized into qualitative and quantitative approaches (Venkatesh et al. 2013, pp. 21–24). Quantitative research represents the systematic investigation of phenomena by collecting data which can be quantified by computational and mathematical techniques. Laboratory experiments or questionnaires are two popular instruments for quantitative research. Scholars typically select the quantitative approach to empirically test hypotheses about relationships between constructs from a theory, since quantitative research is primarily based on deductive reasoning. In contrast, qualitative research is natively more inductive, so the objective is to infer generalized explanations and theories from specific observations. Examples for qualitative research instruments are in-depth interviews or archival research.

In the present thesis, a quantitative approach was employed in order to empirically examine the influences of update-induced issues on app users. For this purpose, hypotheses about post-adoption behavior were deduced from the ISCM. Besides, quantitative research encompasses the subcategories descriptive and experimental research (Williams 2007, pp. 65–66). The former involves the exploration of attributes or factorial correlations of an observable phenomenon in its current state, the latter analyzes the cause-and-effect relationship between variables within a controlled environment. Thus, the adopted research design for this thesis was descriptive ex-post facto by its nature, since the effects of the observed iOS software update already manifested, what excluded the possibility of controlling any independent variables. Another important study design question referred to the choice between primary or secondary data. While primary data is original information acquired from first-hand resources, secondary data is public information provided by someone other than the researcher. As this study was interested in user feedback regarding a very specific and recent event like the mobile operating system update iOS 13, the decision was made to collect primary data

from the App Store. Having answered the fundamental study design questions, the sequential application of concrete methods for the empirical data analysis needed to be determined. In brief, this process aims at making sense of complex data sets by bridging the gap between low-level data and higher-level concepts (Berente et al. 2018, p. 54). This related to techniques for gaining insights about users' perception of the new iOS update that can be extracted from vast amounts of feedback data. So, it was expedient to align the analysis with a suitable framework for knowledge mining.

Knowledge Discovery in Databases (KDD), which was developed by FAYYAD et al., is a widely recognized methodical framework that supports the extraction of information from large volumes of raw data (Fayyad et al. 1996, pp. 37–42). The multi-step KDD process entails a total of five sequential steps: selection, preprocessing, transformation, data mining, and interpretation. First, a target data set is selected on which discovery is to be sought. The second step improves data reliability through cleaning and preprocessing activities. Third, feature selection or dimension reduction techniques convert the data set into the required form for the analysis. In the fourth step, selected data mining algorithms are applied to derive patterns and correlations of interest. Fifth and finally, the mined patterns are interpreted and can be displayed in models or visualizations. The discovered knowledge can be documented for reports or integrated into another system, and, of course, it can be used for hypothesis verification. According to FAYYAD et al., the KDD process allows for multiple iterations if the found patterns are insufficient for the set research objectives (Fayyad et al. 1996, pp. 50–51). They also note the suitability of the KDD framework for mining free-form text data.

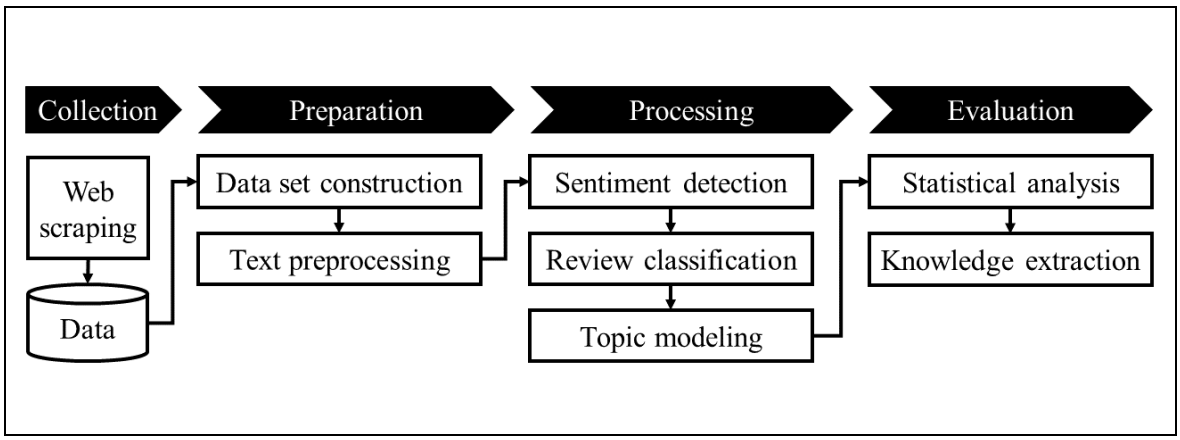


Figure 3: Schematic diagram of study design

Hence, KDD elements were partially adopted for creating the design of the study at hand, which is depicted above in Figure 3. The four main phases collection, preparation, processing, and evaluation are represented as the sequence of black arrows. Beneath each phase, the high-level process flow is displayed. Put concisely, primary data was first collected and

stored with the help of Python-scripted web scrapers. The raw customer reviews then underwent a preparation phase, where the data set construction and text preprocessing tasks were performed. Next, the reviews were ready for processing through data mining algorithms. In more detail, the sentiment of each review text was identified, reviews related to iOS 13 were classified and eventually the latent topics within these user reviews were extracted. Last, the discovered data patterns were evaluated in the fourth phase. Descriptive statistics, correlation analyses, and statistical tests helped with the empirical investigation and extraction of relevant knowledge. Special attention was paid to users' intention of sustained information systems usage as well as its antecedent constructs. Depending on the outcome of the data evaluation, the proposed hypotheses from chapter 3 were either accepted or rejected.

After having briefly summarized the four different phases in the study design at hand, a deeper look into how exactly the phases were carried out is appropriate. Therefore, the succeeding chapters elaborate on the underlying workflow behind each phase.

4.2 Data collection

4.2.1 Sample definition

As outlined in the previous chapter, the study intends to explore the impact of Apple's last operating system update on app users' post-adoption behavior. To assess the iOS patch impact, it was essential to obtain a broad measure of corresponding feedback data from customers. In terms of data source, it goes without saying that the choice fell on Apple's digital marketplace as repository for app reviews. A thorough inspection of the App Store can be found in chapter 2.1.1. The decision to acquire the required feedback data from app reviewer comments and ratings was made during the study design. Being the data object of interest, the format of a user review can vary across different app marketplaces.[2] Reviews on the App Store contain five attributes that are visible to all customers: author, date, review text, rating, and the number of people who found the review helpful (Apple Inc. 2018). The textual content is made up of a title together with a short text, usually consisting of more than one sentence. Customers can rate any app on a scale of 1 to 5 stars. Feedback can only be written by customers who have actually downloaded the application. Aside from the attributes above, additional review features can be gathered from Apple's App Store, such as, internal app ID, country code, or app version number. All these features were received for each collected review. At this point, geographical and temporal restrictions during the data acquisition process are also worth mentioning. Since the text mining methods were confined to reviews in English language, only reviews from both the United States and Great Britain marketplaces were selected. Considering the time limitation, the data collection period was

[2] The elementary structure of customer reviews on the App Store is depicted in Appendix A.1.

defined to span four months, whereby the interval started one month before and ended three months after the initial release of iOS 13, which happened on 19 September 2019.

Even though the research question concerns how the iOS update influenced customers of Apple's ecosystem, which offers millions of applications, it is not feasible to gather reviews from the whole app population due to limited time and computational power. That is why a representative sample was necessary. The constructed sample was based on several criteria. For one thing, it reflected the product diversity in the App Store, and for another, it included applications that attract relatively more feedback because of high popularity among many customers within the ecosystem. From the more than 20 different categories in Apple's portfolio, a subset of 15 categories was chosen. It was ensured that these selected categories accounted for the most app downloads worldwide as reported by current market statistics (Sensor Tower Inc. 2019, pp. 24–27). For instance, Games, Photo & Video, and Entertainment made up the list of top three most downloaded app categories. The majority of downloads were associated with leading developer studios from Facebook, Google, or Amazon. It was assumed that if professional publishers encountered app issues due to the iOS 13 patch, then smaller or independent developers should be at least as heavily affected. Furthermore, the sample included the ten most popular free apps for each of the chosen categories. These top ten lists were manually fetched from the App Store in mid-November 2019. The idea behind this originated from the relationship between high popularity and larger amounts of feedback. More feedback, in turn, implied a higher likelihood of reviews disclosing desired information about the full spectrum of iOS problems. In total, the sample comprised 150 distinct apps.[3] Below, Table 1 shows a limited extract of this sample with the names of the top three ranked apps within each category.

Category	**App ranked first**	**App ranked second**	**App ranked third**
Books	Audible	Amazon Kindle	Wattpad
Business	Indeed Job Search	ZOOM Cloud Meetings	ADP Mobile Solutions
Education	Photomath	Google Classroom	Remind
Entertainment	TikTok	Netflix	Hulu
Finance	Cash App	Venmo	PayPal
Food & Drink	DoorDash	Uber Eats	Grubhub
Games	Call of Duty Mobile	Photo Roulette	Rescue Cut
Health & Fitness	Calm	Reflectly	Flo
Lifestyle	Google Home	Tinder	OnMyWay
Photo & Video	YouTube	Instagram	Snapchat
Productivity	Gmail	Google Docs	Google Drive
Shopping	Amazon	Walmart	Wish
Social Networking	Facebook	Messenger	WhatsApp Messenger
Travel	Uber	Lyft	Yelp
Utilities	Google Chrome	Google	Fonts

Table 1: Extract from mobile app sample

[3] The complete list of all 150 mobile apps of the sample can be found in Appendix B.2.

Whenever a sample is taken in order to make inferences about a population, awareness towards threats to external validity is mandatory. What can lead to biased data is a sampling method that systematically favors some outcomes over others. MARTIN et al. presented the app sampling problem for app store mining which can occur when only a subset of apps is studied by selecting, for example, the most recent or most popular apps (Martin et al. 2015, pp. 132–133). Nevertheless, they found that correlation analysis and topic modeling were less sensitive to biasing effects. Hence, regarding the defined sample and used methods in the present thesis, the app sampling problem was not a major risk. Still. it was obviated to make vast generalizations for the entire app population.

4.2.2 Data scraping process

In the course of the preceding sample definition, general conditions of the collection process were set, for example, temporal, geographical, or content-related specifications. Following the exact description of what data were required, this chapter addresses how the data collection was executed. The compiled sample list of 150 applications constituted the starting point. Information about app feedback is publicly available on the App Store. Web mining is thus common practice for collecting user reviews. The term web mining, first coined by ETZIONI in 1996, refers to the application of data mining techniques to discover and extract information automatically from web documents or services (Etzioni 1996, p. 65). However, compared to other app markets like Google Play, Apple is more restrictive when it comes to allowing software agents to gather data from its webpages. Accessing the App Store through a web browser gives merely a constrained preview of user feedback. As an alternative source, Apple provides web feeds based on rich site summary (RSS) documents that hold information on iTunes hit lists, developer news, mobile app reviews and much more. Therefore, pertinent RSS feeds were queried with the help of specifically programmed web crawlers.

Scrapy is an open source web crawling and web scraping framework written in the programming language Python. It is designed to crawl and extract structured data from webpages (Scrapinghub Ltd. 2019). Merging efficiency, performance, and an excellent developer documentation, it is a popular solution in the industry. Built-in data export formats cover CSV, JSON and XML. Owing to its easy expandability, Scrapy has the option to add custom middleware or data pipelines. Having only a compatible Python version and a virtual environment as prerequisites, the setup was fairly simple as well. In light of these advantages, the framework was a proper choice for the data collection procedure, whereby the versions Python 3.8 and Scrapy 1.8.0 were used. The scraping process can be divided into two steps, systematically finding and downloading webpages, and then parsing information from the pages' source code (Scrapinghub Ltd. 2018). Scrapy's architecture incorporates an engine controlling the data flow between various components, including a scheduler, a downloader

and customized classes for data extraction, also known as spiders or bots.[4] The general data flow can be summarized as follows. At first, the Scrapy engine gets the initial Hypertext Transfer Protocol (HTTP) requests from the spider and then schedules the requests in the scheduler. Once the scheduler returns subsequent requests to the engine, they are passed through the downloader. After the download finishes, the downloader sends a response to the engine. Next, the engine forwards the response to the spider, which thereupon processes the downloaded webpage and returns scraped items to the engine along with new HTTP requests. Ultimately, the engine sends the processed items to the data pipeline and asks the scheduler for new requests. This cycle repeats until there are no requests left. Accordingly, the programmed Python module started by creating HTTP requests targeted at Apple's customer review web feed. Query parameters concerning the app ID, time period, and App Store country code were passed through the target Unified Resource Locators (URLs) in a loop. To systematically scrape data in the feed format, custom spiders were programmed. Since the downloaded RSS documents consisted of XML-formatted plain text, the spiders exploited Scrapy's XPATH selector mechanism. XPATH is a language for selecting nodes and processing items in XML documents. Scripted XPATH expressions enabled the spiders to parse the RSS pages. The retrieved customer reviews were stored in CSV format.

Although the feed scraper module routinely fetched data for all 150 sampled applications, the data was insufficient due to a limitation of Apple's web feed. In fact, the feed solely stored records about the 500 most recent customer reviews for a single application excluding past reviews beyond this limit. This turned out to be a problem for apps with high feedback rate because some historical data within the four-month collection period were missing. To acquire the needed historical records, a software agent was programmed that systematically imported data out of the API from Appfigures, which is a mobile app store reporting platform (Appfigures Inc. 2019). For getting access to the API, a 14-day free trial account was registered. Despite the rate limit of 1,000 API requests per day in the test version, it was manageable to obtain the remaining historical data. The programmed software agent worked in a similar way like the feed scraper module. First, the software agent authenticated itself to the Appfigures server with a client key from the trial account. Then, it proceeded to send API requests, downloaded the responses in JSON format and stored the information after the parsing operation in a CSV file.

As final outcome of the data acquisition, two raw data sets were scraped from two distinct origins. In combination, they included all required customer reviews for the 150 sampled apps in the defined four-month time span. Still, dissimilarities between the data sources were reflected in the data sets themselves, thereby entailing the need for further preparatory steps.

[4] The architecture of Scrapy is graphically illustrated in Appendix A.3.

4.3 Data preparation

4.3.1 Data set construction

Within the KDD process, paramount importance is attached to the data preparation phase. It encompasses various preprocessing activities for transforming raw data into a structured format that is amenable to data mining procedures (Fayyad et al. 1996, pp. 39–41). Some mentionable operations include, for example, variable selection, noise reduction, or handling missing values. Raw data are often erroneous, inconsistent, or incomplete. These deficiencies were taken into account when the two scraped data packages were merged for the construction of one cohesive data set. Stemming from separated data origins, namely Apple's web feed and Appfigures' database, the act of harmonizing proved to be obligatory. Data harmonization focuses on integrating data of varying qualities, types, and naming conventions. To ensure proper data quality, which after all determined the successful analysis, bringing together the scraped data sets was handled through a Python module specifically developed for preparation purposes.[5]

One function in the referred module was dedicated to data set fusion. Its functional architecture emulated the paradigm of load, transformation, and extraction (ETL). This process is widely spread in data warehousing, where data coming from heterogenous sources are restructured and stored (March and Hevner 2007, pp. 1032–1033). The first step in ETL involves extracting all required data from their source systems. During the transformation step, the data are prepared for analysis through aggregation, cleansing, deduplication, and other preprocessing tasks. In the last ETL step, the ready-made data is loaded into the end source. As delineated in the following paragraph, these three steps were programmed into the function devoted to data set merging, though in a slightly adjusted manner.

First and foremost, raw data from the two CSV files were read in and converted into data frames by using the open source software library pandas. Pandas is a powerful, versatile Python package that provides custom data structures and operations (McKinney 2010, p. 51). These implementations supported both the data manipulation as well as the information analysis throughout the course of this study. After the data import, the feature sets were aligned. This implied the enforcement of fixed naming conventions. Identical features were given the same feature names. Aside from renaming, certain attributes had to be deleted because they either were redundant or existed only in one data set. In addition, the scripted function ensured the unification of different data formats for identical features, such as the date format or app version numbering. For instance, in one data set the app version number contained trailing zeros, while in the other there were none. That is why trailing zeros inside

[5] For a structural overview of all developed Python modules, refer to Appendix A.2.

the app version details were truncated across all records. Subsequent to the feature alignment, a horizontal concatenation of both data frames took place. This step resulted in the creation of a single data set with rectified properties, although it simultaneously introduced duplicates. Occasionally, the exact same app review of an author appeared multiple times, yet with minimal variations in the timestamp. To detect and filter out double records, a custom pandas function was applied. Next, the penultimate transformation step joined the data frame with the app sample list by means of the app ID in order to add extra columns, such as app category, developer, and rank. Lastly, the newly constructed data set was loaded into a CSV file, which completed the data set construction phase.

4.3.2 Text preprocessing

Apart from the function described above, a second preparative function devoted to textual data was implemented as preliminary step before natural language processing (NLP). The discipline of NLP concerns itself with the development of computational algorithms that enable machines to process and understand human language (Panichella et al. 2015, p. 282). A major goal in NLP is to extract meaning from spoken or written language. In connection to answering the research question of this thesis, state-of-the-art NLP techniques were used to analyze voluminous textual data inside customer reviews. Typically, feedback comments are not well-written, but rather contain misspelled, repetitive, and filler words. As a matter of fact, many comments are written on limited keyboards of mobile phones. The relatively younger age group of users might be another reason. Consequently, textual data preparation is especially vital when dealing with mobile app reviews. To reduce as much noisy, uninformative text elements as possible, a whole series of preprocessing tasks were carried out.

The creation of a new attribute named document, which concatenated title and content for every individual review, marked the beginning. This way the original review text was preserved, whereas only the text data copied into the document attribute underwent preprocessing. Taken together, the document attributes for all records represented the corpus of documents to be analyzed. It became apparent that some customers composed their feedback in Arabic or Chinese language in spite of the restriction to American and British App Stores. Regular expressions helped to filter out instances with Arabic or Chinese letters, saving time and resources in foresight of the computationally intensive text preprocessing tasks.

Thereafter, the Python script traversed all documents to check for punctuation marks. Punctuation removal is a frequently recommended preprocessing step, since punctuations do not convey any significant semantics. The same holds true in respect of special characters. A custom string which enclosed these unwanted text elements served as search input. If any instances were identified in a document, they were replaced with a whitespace character.

Generally, it is advised to remove digits for noise reduction as well, but this was not applicable for the planned analysis because numbers could stand in relation to a distinct mobile operating system version number like iOS 13.1 or iOS 13.2.2.

Further, converting all review texts to lowercase was another simple but highly effective preparatory step. Many customers excessively capitalize either the first letter or the entire word to emphasize certain words. However, casing does not change the general meaning of a given word. In this context, lowercasing also reduces sparsity and vocabulary size, since identical words with diverse cases map to the same lowercase form. It was thus crucial for improving textual data consistency.

Stop word removal belongs to the most popular preprocessing steps across a wide range of NLP implementations. Stop words are commonly used words in a specific language. Articles, pronouns, and prepositions count as examples. A list of English stop words was imported from the Python NLTK library, which is a leading open source platform offering several tools regarding NLP (Bird et al. 2009, p. 9). To give but a few concrete examples, the list included words like “and”, “the”, “is”, or “me”. The idea behind removing stop words is that uninformative elements are deleted, while the share of relevant top-specific terms in a document increases. Consequently, the number of features representing a specific customer review was trimmed down to a few truly significant words.

Besides lowercasing, the act of stemming is a supplementary text normalization technique. Stemming reduces words to their root form. Relatively simple algorithms cut off suffixes from words to retain the word stem. This may result in stems that are not actual words, but just chopped of terms. Overstemming occurs when two distinct words are stemmed falsely to the same root. An alternative operation to stemming is called lemmatizing. Unlike the former, lemmatization groups together inflected forms of a word to its dictionary form based on lexical knowledge. Lemmatized words are therefore always valid words. As lemmatization produced more readable results, aiding the interpretation of modeled topics, NLTK’s default WordnetLemmatizer was applied to the text corpus.

For the sake of regularity, the last preparatory action eliminated redundant whitespace characters so that all text elements were separated from each other by a single whitespace. By way of illustration, a user review before and after preprocessing is cited below in Table 2.

Raw document	**Preprocessed document**
App keeps CRASHING!!! After updating my iPhone to IOS 13.2.2, I have noticed the app takes longer to load and crashes more when trying to access it, hopefully, the app can be updated to accommodate this issue :(	app keep crashing after updating iphone ios 13.2.2 noticed app take longer load crash trying access hopefully app updated accommodate issue

Table 2: Comparison of raw and preprocessed review text

4.4 Data processing

4.4.1 Sentiment detection

The readily prepared data set constituted the basic precondition for knowledge discovery. With the ensuing data mining operations, hitherto unknown and useful patterns were exposed. Most notably, unstructured data inside the customer review texts, which were cautiously preprocessed, embodied a main information source of great potential. In the field of NLP, one prevailing approach to making sense out of textual feedback and customer attitudes is sentiment analysis (Chen et al. 2014, p. 768; Panichella et al. 2015, p. 284). It is responsible for determining the semantic orientation of provided source material. The expressed affect or mood can be quantitatively assessed through so called sentiment scores. In contrast to conventional customer ratings, sentiment scores provide a more fine-grained annotation. From the calculational perspective, sentiment scores are computed using either lexicon-based or machine learning approaches. The latter approach builds classification models that learn from training data what kind of sentiment is associated with various textual features. Lexicon-based approaches require a dictionary of positively or negatively connotated words. A summing function evaluates the overall sentiment of a document by taking into account the individual sentimental values assigned to each of the words. Some hybrid techniques even blend aspects of the two main approaches.

In the present thesis, sentiment detection was conducted with the Valence Aware Dictionary and sEntiment Reasoner (VADER) package, a lexicon and rule-based sentiment analysis tool (Hutto and Gilbert 2014, p. 216). Among numerous reasons for choosing VADER, its aptitude in respect of microblog-like contexts is to be emphasized. Accordingly, the VADER lexicon is especially attuned to sentiment expressions in product reviews or social media comments, which are often characterized by abbreviations, slang, and informal language. VADER includes a list of sentiment-bearing lexical features relevant to this domain. These lexical features are combined with five general rules considering grammatical and syntactical conventions that humans use when accentuating sentiment intensity. Due to this mechanism, VADER does not require a large training data set, yet, as HUTTO and GILBERT demonstrated, it performs very well in direct comparison with other highly regarded sentiment analysis solutions (Hutto and Gilbert 2014, pp. 224–225). In sum, the above stated reasons justified the suitability criteria of the tool.

As previously handled, a new Python module for data processing was created. After importing the VADER package, the function dedicated to sentiment detection loaded the prepared data set into a data frame structure. Then, the script instantiated an object of the SentimentIntensityAnalzyer class. An implemented method in this class takes a string argument,

calibrates its sentiment orientation and returns a dictionary with corresponding polarity scores. The output dictionary not only contains the proportional categorization into positive, neutral, or negative but also a compound score summing up all lexicon ratings. The positivity score describes the overall sentiment polarity on a scale standardized from 0 to 1. Values close to 1 reflect a high percentage of positivity, whereas 0 denotes complete absence of positive sentiment. Consequentially, the programmed function iterated over each review record to analyze its sentiment and stored the polarity scores in a newly added attribute column. Ultimately, the data set with annotated sentiment was saved.

4.4.2 Review classification

In order to investigate the iOS update-induced impacts on mobile applications, over half a million customer feedbacks were collected. Given this background, the fundamental question arose whether an arbitrary review referred to iOS 13 or not. A binary classification into either iOS update related or unrelated presented the solution. Without doubt, the sheer number of review texts called for computer-aided categorization procedures. Thus, methods were employed from the research field of machine learning, which lies at the intersection of artificial intelligence, computer science, and statistics. Machine learning focuses on developing systems capable of learning patterns and making predictions from data (Fayyad et al. 1996, pp. 43–44). For the task at hand, machine learning algorithms that classify data items into one of multiple predefined classes were pertinent. However, the plethora of options to choose from made the model choice non-trivial. A two-step approach was used to guide the review classification process, where the comparison of the two differently classified subsets was conducted as an additional data validation check.

In a first step, customer reviews with a very high certainty of being related to the recent iOS update were preselected by scanning through all data records. A simple text search was carried out to identify co-occurrence of the term iOS and number 13 in the same document. Regular expressions allowed specifying character sequences for matching conforming search patterns. Thereby, all possible constellations between the textual elements iOS and 13 were covered. Since the term iOS accompanied by the number 13 turned out to be a quite particular combination, an intermediate inspection of the results attested an extremely high classification accuracy. Despite of its exactness, the simple text search strategy ignored many reviews that implicitly mentioned iOS 13. For example, customers often referred to it as the newest firmware update, upgraded operating system or recent iPhone version without mentioning iOS 13 explicitly. Hence, more sophisticated classification techniques from the machine learning domain were also put to the test.

The second step aimed at using machine learning algorithms to learn hitherto unidentified text features that were strongly correlated with the topic iOS 13. This attempt sought to create a bigger subset of iOS feedbacks through detecting relevant instances that were evidently more difficult to classify and thus overlooked in the prior text search. Python's scikit-learn package is one of the most widely used open source libraries for machine learning and predictive modeling (Pedregosa et al. 2011, p. 2825). It integrates state-of-the-art machine learning algorithms, including support for classification, regression, cluster analysis, and model selection. Unlike other major machine learning toolkits in Python, scikit-learn incorporates compiled code for efficiency, what gives it a significant advantage in performance. Owing to its consistent, task-oriented interface, it enables an easy comparison of implemented methods. Its minimal dependencies and an excellent documentation count as additional reasons why scikit-learn is used in both academic and commercial settings.

Concerning the review classification, scikit-learn offered supervised and unsupervised learning methods (Pedregosa et al. 2011, pp. 2827–2829). Supervised methods try to find relationships between the feature set and predefined classes. In other words, they are designed to infer a classifier function by learning from labeled data examples with indicated classes. This learned function takes in unseen inputs and outputs the predicted class labels. As distinguished from supervised models, unsupervised machine learning does not presuppose any training data (Aggarwal and Zhai 2012, pp. 5–6). Its main goal is to capture hidden structures in unlabeled data. Association rule mining, clustering, or dimensionality reduction are three popular techniques for unsupervised machine learning. To predict if a customer review refers to Apple's iOS patch, the decision fell on supervised machine learning methods because the task could be cast as binary classification problem. Besides, labeled data records were already available from the earlier conducted text search with regular expressions. These labeled reviews about iOS 13 served as training examples.

The model selection entailed a thorough comparison of several candidates. Three universally recognized methods were elected from among the multitude of text classification algorithms, namely naïve Bayes, logistic regression, and random forest. The naïve Bayes classifier is commonly used for establishing a first baseline. It is a simple generative classification algorithm that models the document distribution for each class in a probabilistic manner (Aggarwal and Zhai 2012, pp. 181–182). Following Bayes' theorem, it calculates the posterior probability of a class by exploiting the distribution of words in a given document. The calculation is based on making the simplifying assumption of stochastic independence between word features in each class. Assuming this conditional independence is hardly ever true in practice, which is why it is titled naïve. In spite of its strong independence assumption, naïve Bayes achieves good classification performance when the feature dependencies cancel one another out. A multinomial model is typically used for document classification, whereby

word frequencies are captured in a so-called bag-of-words vector representation, ignoring grammar or word positions. Bayes' theorem is applied to the multinomial word distribution to compute the posterior probability of the class for a given document. Mathematically speaking, the posterior probability of the observed feature vector $x_1, x_2, \dots, x_n$ belonging to a particular class C_j can be estimated as the product of individual feature probabilities conditioned on class C_j multiplied with the prior class probability $P(C_j)$ divided by the prior probability of the predictor variables $P(x_1, x_2, \dots, x_n)$. Since the latter is constant for all classes, the denominator is often neglected in the calculation. Out of all the different classes, the label of the most probable class is assigned to the document.

$$P(C_j|x_1, x_2, \dots, x_n) = \prod_{i=1}^{n} P(x_i|C_j) \cdot \frac{P(C_j)}{P(x_1, x_2, \dots, x_n)}$$

Logistic regression, being the discriminative analog of naïve Bayes, was chosen as a second option. While generative algorithms must assume some conjectures about the underlying data structure, such as feature independence in naïve Bayes, discriminative algorithms make fewer assumptions (Ng and Jordan 2002, pp. 842–843). Contrary to generative classifiers that try to compute the actual class distribution, discriminative ones compute the decision boundary between classes. The mathematical formula of the binary logistic regression model is expressed below. In the case of a linear classifier like logistic regression, the decision boundary is determined by a linear combination $\beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \dots + \beta_n \cdot x_n + \varepsilon$ of n different features plus an error term ε (Aggarwal and Zhai 2012, pp. 196–197). Provided a set of training data, the model is trained through adjusting the β parameters to maximize the conditional likelihood of labels C_j given the corresponding $x_1, x_2, \dots, x_n$ observations. As denoted in the binomial logistic regression equation below, the logistic sigmoid function is applied to transform the output into a probability value between 0 and 1. To map the resulting value to a binary category, a classification threshold of 0.5 is set by default. Having calculated the posterior class probability, the other class gets assigned the complementary probability $1 - P(C_j = 1|x_1, x_2, \dots, x_n)$. Even though logistic regression can be generalized to multiclass problems, the binary model was sufficient for the planned review classification.

$$P(C_j = 1|x_1, x_2, \dots, x_n) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \dots + \beta_n \cdot x_n + \varepsilon)}}$$

Random forests completed the selection of three classification methods. This family of ensemble machine learning algorithms has proven to be especially suited for dealing with text categorization (Fernández-Delgado et al. 2014, p. 3133). The random forest classifier, like its name suggests, creates an ensemble of decision trees where each tree is constructed from randomly selected training data subsets. By aggregating the predictions from the decision tree ensemble, the final class prediction is derived. Decision trees are capable of modeling

one complex decision into a sequence of primitive rules. They are designed to perform hierarchical division of the training data space, in which conditional control statements with respect to an attribute are traversed recursively (Aggarwal and Zhai 2012, pp. 176–177). The splitting objective is to produce more homogeneous subspaces than the original data space in terms of their class distributions. In the context of text classification, splitting conditions are usually about the absence or presence of specific terms. To quantify discriminative abilities of words, the implemented tree algorithm used the Gini-index. As mathematically expressed below, the Gini-index is calculated by summing the squared fractions of class label presence $p_j(w)^2$ for word w across the K different classes (Aggarwal and Zhai 2012, p. 168). Gini-index values $Gini(w)$ lie in the range $(1/K, 1)$, whereby higher values indicate greater discriminative power of word w. Given an exemplary document, decision trees apply a sequence of rules at their nodes to divide the data space until a leaf node is reached. Depending on the majority class inside the leaf node, the respective label gets assigned. Unfortunately, out of all machine learning algorithms, decision trees are amongst the most susceptible to overfitting. The latter occurs when a model captures random noise rather than to generalize from the training data. Pruning nodes to reduce complexity is a way to mitigate this problem. Further, irrespective of pruning, the random forests classifier is robust against overfitting due to its inherent randomness and voting mechanism as ensemble method because picked up noise averages out across various decision trees.

$$Gini(w) = \sum_{j=1}^{K} p_j\,(w)^2$$

Returning to the two-step review classification process, the second step was devoted to training the three discussed classifiers. A programmed Python function sampled roughly 25,000 records from the preprocessed data. Labeled iOS 13 reviews that resulted from the preceding text search were integrated in this training data set. Then, all text documents were converted into numeric vectors to enable mathematical computation. An aforementioned technique for mapping words to vectors, also known as word embedding, is the bag-of-words approach, where documents are represented as multidimensional vectors with word counts. But instead of relying on normal enumeration, another weighting schema named term frequency-inverse document frequency (TF-IDF) was preferred (Aggarwal and Zhai 2012, pp. 389–390). TF-IDF weights measure how relevant a word is to a document in a document corpus.

$$w_{s,t} = tf_{s,t} \cdot log\left(\frac{D}{df_s}\right)$$

According to the TF-IDF formula above, the word weighting factor $w_{s,t}$ is calculated by multiplying the frequency of the term s in the document t with the logarithm of the inverse document frequency, computed as the total number of documents D divided by the number

of documents containing s. Put differently, word importance is proportional to the number of times it appears in a document, but is offset if the frequency of the word in the document corpus is high. For converting review texts into a matrix of TF-IDF features, scikit-learn's TfidfVectorizer function was adopted. The TfidfVectorizer function also allowed the extraction of word n-grams, which encompassed all combinations of adjacent words of length n from the source text. Through including both unigram and bigrams, some degree of contextual and word-order information was retained. As a result, the calculated TF-IDF weights were incorporated in a two-dimensional word-document matrix whose rows represented document vectors and columns represented all word features in the corpus. This matrix was split randomly with ratio 80:20 into training and test subset, respectively. Next, naïve Bayes, logistic regression, and random forest classifiers were fitted on the training records and afterwards tested on the validation subset. A comparison between the three learned models assessed performance indicators, such as accuracy, precision, recall, or the F_1-score. While accuracy measures the overall correct discriminations, precision reflects the percentage of correct instances among all instances classified as positive (Aggarwal and Zhai 2012, p. 33). Recall measures the percentage of correct instances retrieved from all positives. The F_1-score takes the harmonic mean of precision and recall. Finally, predictions of the best performing model were stored in the data set, which consequently listed two class labels from both the regular expression text search and the machine learning approach.

4.4.3 Topic modeling

Aside from the executed document classification, a successive data mining task concentrated on corpus summarization. Outcome of the classification process was a subset of customer reviews with feedback regarding iOS 13. However, the sheer number of reviews made a manual content analysis impossible. Computer-aided knowledge extraction was necessary. That is why advanced NLP techniques were employed to provide summary insights into the overall content of the review text collection. A practical method for this purpose is topic modeling (Aggarwal and Zhai 2012, pp. 131–132). The general idea behind topic modeling is to discover hidden semantic patterns inside large bodies of text. Because of its connection to clustering as well as dimension reduction, topic modeling is considered as unsupervised machine learning. Unsupervised techniques can transform documents into new representational ways, which reveal their innate structures and interrelations. Topic modeling groups documents into thematic clusters based on information about document similarity. Comparable documents are associated with the same clusters, though, since topic modeling is a probabilistic approach, soft clustering is used, wherefore one document can be assigned to multiple topics. Each document has a membership probability of belonging to a certain thematic cluster and each cluster is a function of all original text features. Considering each topic as a dimension, clustering induces a low-dimensional view, what supports analysis and

interpretation. In other words, topic modeling integrates soft clustering with dimension reduction by linking documents with a predetermined number of latent topics.

In the thesis at hand, topic mining was performed with the latent Dirichlet allocation (LDA) algorithm. LDA, first introduced by BLEI et al. in the year 2003, is a probabilistic generative model for documents (Blei et al. 2003, pp. 993–996). It is a Bayesian model and derives its name from having a Dirichlet prior. Per definition, the model treats each document as a distribution of a preset number of T topics. Each topic, in turn, is treated as a multinomial distribution of words. In a further step, it is assumed that documents are generated by selecting a topic set and then for each topic a set of words. This generative process is depicted in the plate diagram in Figure 4 below.

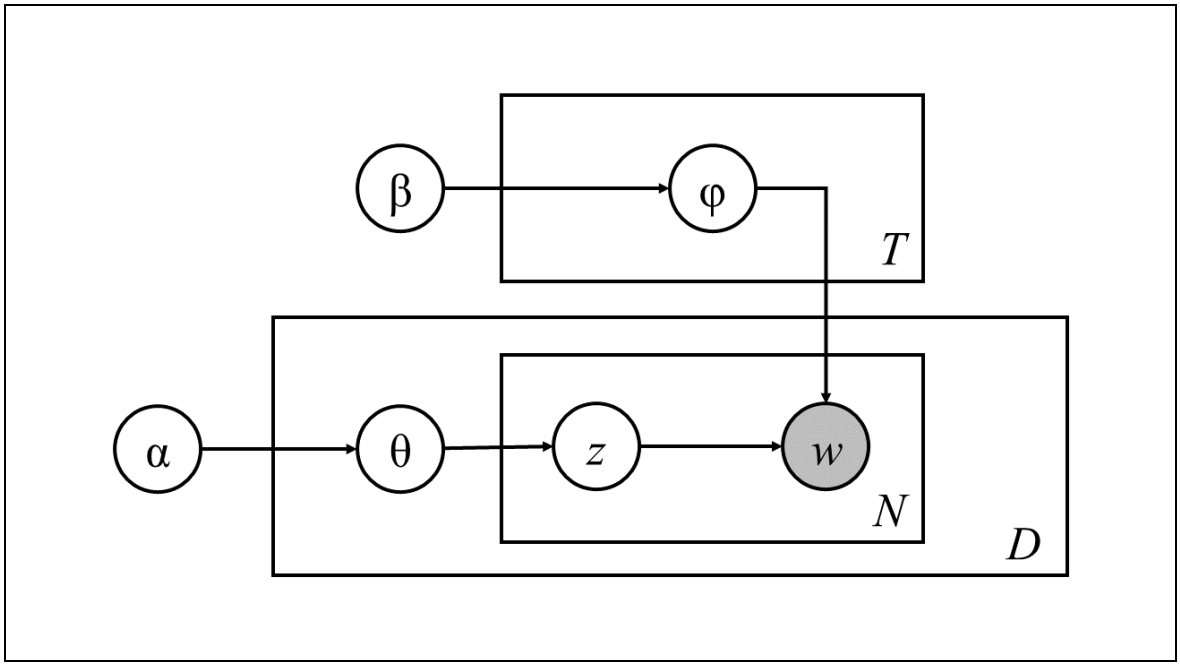


Figure 4: Plate notation for latent Dirichlet allocation (Blei et al. 2003, p. 1006)

Initially, the chosen Dirichlet parameters α and β govern the per-document topic distribution and per-topic word distribution, respectively. In plate notation, rectangles symbolize repeated variables. Thus, a word distribution φ exists for each of the T topics and in the same manner each of the D documents has its own topic distribution θ. A single document consists of N words in total, thereby, the hidden topic variable z appears N times. Eventually, each document is generated by drawing N words w from the multinomial distribution $p(w|z)$. It is worth noting that posterior inference works backwards reversing the generative process such that hidden model parameters are predicted using text data of the corpus (Aggarwal and Zhai 2012, pp. 142–144). As words are the only known piece of data, the observable word variable w is shaded in the diagram. Training an LDA model involves estimating ideal parameters, under which the likelihood of generating the training documents is maximized. Therefore, LDA results in patterns of terms which frequently co-occur with each other in

documents. Coherent topics are identified amidst these patterns. Moreover, another benefit of LDA is its capability of resolving polysemy and semantic ambiguity. For example, a term appearing in multiple topics can have distinct meanings, however, LDA disambiguates polysemous terms by leveraging contextual information provided that the topics are associated with other words in the document. Also, LDA favors learning a smaller number of broad topics rather than a large number of overly specific ones, which was appropriate for the intended high-level analysis of update issue categories. Taken together, these advantages predestined LDA as the optimal topic modeling algorithm for this thesis.

Concerning the data processing script, the NLP toolkit gensim was imported (Rehurek 2019). It provides various topic modeling scripts along with an implementation for LDA. To begin with, categorized iOS 13 feedbacks in their entirety were loaded into a list of texts. A loop function subsequently tokenized each text document into a list of words. If adjacent terms co-occurred at least 15 times throughout the corpus, bigrams were formed. Additional preprocessing was not required due to the earlier text preparation. Next, a function from gensim constructed a dictionary with a mapping between words and integer identifiers. Extreme cases, like tokens that were present in less than five documents, were filtered out from the vocabulary. As LDA relies on term frequency vectors, the corpus and dictionary were combined to construct a bag-of-words representation. Several LDA models with a varying number of fixed topics were trained on the bag-of-words corpus. The information retrieval indicators coherence and perplexity helped with scoring each model (Aggarwal and Zhai 2012, pp. 149–150). Coherence measures the degree of semantic resemblance between top words of a topic. Perplexity indicates how well a probabilistic model predicts a sample. Ultimately, the output of the best topic model was captured for the ensuing empirical analysis.

4.5 Data evaluation

In accordance with the study design, the last step of the knowledge discovery process was reserved for data evaluation. During the previous processing phase, a series of data mining algorithms were applied, such as review classification, sentiment detection, and thematic corpus summarization. The data evaluation step referred to interpreting those mined data patterns (Fayyad et al. 1996, pp. 41–42). It involved statistical analyses, quantitative measures, and visualization techniques. Two chapters were dedicated to data evaluation. Chapter 5 provides a descriptive commentary that discloses empirical findings as well as analytical insights, underpinned by graphs and tables. Chapter 6 elaborates on the findings and explains uncovered correlations, supported by personal interpretations and related literature. On this evidence basis, the three research hypotheses were discussed and verified. A brief overview of the data evaluation procedure is outlined in the following passage.

The procedure started with evaluating the document classification, since all ensuing analyses built upon the categorized review data. After comparing performance metrics of all three classifiers, the best model was selected. The predictions from this machine learning model were benchmarked against the matches of the regular expression text search that served as training data. Categorized instances of these approaches were systematically examined to ensure data validity of both subsets. Then, classified iOS 13 feedback texts from both subsets were merged into one data set. Descriptive statistics were used to show the characteristics and distribution of iOS-related comments across various dimensions, such as app category or popularity rank. Thereafter, another evaluation step analyzed sentiment and rating scores to ascertain the detrimental influences of the operating system update on user satisfaction. It was searched for negative review polarity and declines in star ratings. In particular, biserial correlation coefficients were computed to quantify the relationship between negative customer feedback and iOS 13 references. Furthermore, the evaluation procedure focused on cluster analysis in the course of choosing the best LDA topic model. Coherence and perplexity scores of topic models with a varying number of topics were assessed. After the best model was chosen, the extracted topics were examined. Documents with strong thematic similarity were grouped in order to form distinct update issue types. Each identified issue type was studied in depth with quantitative and qualitative analyses. The discoveries were used to prove the manifestation of separable update issue categories in customer feedback. For each established issue type, adverse impacts on users' continuance intention were investigated. Major declines in ratings or sentiment polarity were scrutinized. In this regard, a Kruskal-Wallis test, sometimes referred to as one-way analysis of variance on ranks, was carried out in order to test if the negative impact of different update issue categories varied in terms of magnitude. Along with the test statistics, p-values were calculated to determine the significance of the results, whereby a significance level of $\alpha = 0.05$ was designated as threshold for all tests.[6] Eventually, the series of evaluation results culminated in the concluding chapter 7 with a summary of the key findings and a statement in respect of theoretical and practical implications.

[6] Significance levels: significant at $p \leq 0.05$; very significant at $p \leq 0.01$; highly significant at $p \leq 0.001$

5 Empirical analysis and results

The present chapter deals with empirical analyses of the processed reviews. As an essential part of the data evaluation, it describes the results of applied data mining methods. Firstly, three candidate models are assessed regarding their classification performance. Subsequent to the model selection, a statistical summary and inspection of the classified review data is provided. Another subchapter continues with a temporal analysis of rating and sentiment. Lastly, extracted update issue types from topic modeling are evaluated in detail.

5.1 Assessment of classification models

A direct comparison of multinomial naïve Bayes, logistic regression, and random forest classifiers was done in consideration of model goodness. Through the regular expression text search, 3,416 reviews with iOS 13-related content were labeled, which made up a small share of the total of 646,580 scraped records. Thus, oversampling was conducted in order to avoid extreme class imbalance. The final training data set consisted of 25,000 documents, including all 3,416 labeled iOS 13 instances. An estimator function from scikit-learn helped during the training phase, where the data set was split in k sets and each model was fitted k times on different subsets, with one set held out each time for testing (Pedregosa et al. 2011, p. 2828). This procedure, also known as k-fold cross validation, was initiated setting $k = 5$. Below, Figure 5 depicts the cross-validation outcome. Black dots mark the model accuracies.

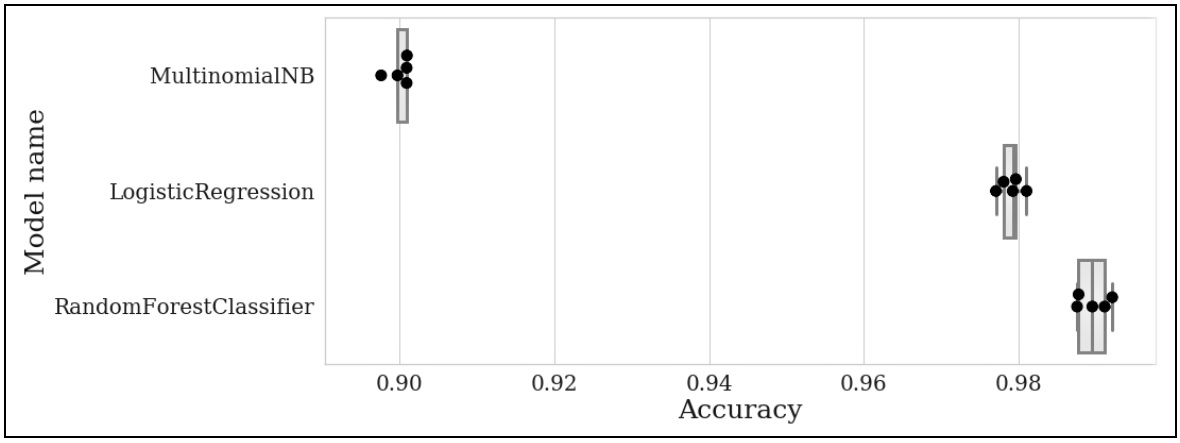


Figure 5: Boxplots of cross validation accuracy scores

In Figure 5, discrepancies between the three models become apparent. In comparison with naïve Bayes (NB), logistic regression as well as random forest classifiers had superior classification accuracies, though the Bayesian model showed less variability. This was partly attributed to the bias-variance trade-off. High bias emerges from oversimplifying model assumptions, whereas high variance models make less assumptions and pay more attention to training data. Naïve Bayes' strong independence assumption implied a lower variability, but it could not accurately represent the data leading to worse accuracy. Another reason was

that, given enough training data, discriminative models, like logistic regression or random forests, tend to outperform generative models. Previous research demonstrated this phenomenon across many situations (Ng and Jordan 2002, p. 841). Yet, accuracy scores alone were not enough for the model selection. Table 3 lists supplementary performance measures.

	Multinomial NB	**Logistic regression**	**Random forest classifier**
Accuracy	0.90	0.98	0.99
Precision	0.97	0.99	0.99
Recall	0.28	0.86	0.93
F_1-score	0.43	0.92	0.96

Table 3: Performance metrics for classification model assessment

The metrics confirmed the inferiority of naïve Bayes. In the test data, the majority of actual iOS 13 feedbacks were not recognized by the Bayesian model. Hence, its recall score of 0.28, the ratio of correctly classified iOS 13 reviews, was roughly three times lower than the score of logistic regression. Although naïve Bayes is often proposed in text classification literature, it turned out to be suboptimal for the classification problem at hand. In contrast, logistic regression and random forests were both very good in terms of model performance. Accuracy, precision, and F_1-score happened to be similar in both models. Still, the random forest classifier achieved a 7 percentage points higher recall value. From out of 665 iOS 13 comments in the test set, it retrieved 618 comments and logistic regression recognized 573. These differences widened after the two learned models predicted the full data set. While random forests missed 47 among 3,416 relevant instances, logistic regression failed to categorize 351. The former had also a 12 percentage points better precision than the latter when predicting the entire data set. Additionally, false positive predictions stood in the center of interest because they included reviews which were potentially related to the recent iOS patch without mentioning iOS 13 explicitly and thereby could not be labeled by the prior text search as such. The logistic regression and random forest classifiers predicted 1,271 and 714 false positives, respectively. A manual check exposed that the more inaccurate logistic regression overgeneralized. The 714 false positives from the stricter random forest model were more promising. By way of exemplification, Table 4 lists a few unprocessed user comments of false positive predictions stemming from the random forest classifier.

App	**Date**	**Review content**
Audible	23.10.2019	Audible won't load now on my 11 Pro running 13.1.3 :(
Capital One	24.09.2019	Great app on every level, however the widget that provides your account balance has stopped working after upgrading to 13.0 and later 13.1
Fitbit	17.12.2019	Ever since the new IOS came out I haven't been able to receive texts on my Fitbit Alta HR. It's been far too long for this issue to not be resolved.
Instagram	02.10.2019	Recently Apple has included dark mode in their new iOS update which has changed the themes in many applications. It would be cool to add dark mode.
Youtube	16.10.2019	The app has become 95% non functional since the 13.1.3 update

Table 4: False positive predictions of random forest classifier

The false positive predictions of the random forest classifier demonstrated its ability to generalize. It successfully picked up text patterns correlated with Apple's thirteenth operating system update. During the training phase, the algorithm found associations in respect of iOS 13-related version numbers, iPhone models, or specific application issues. The same held true for prevalent words in the software update context. Owing to the easy interpretability of decision trees, learned associations were observable within the individual tree structures.

```
feature_names = tfidf_vectorizer.get_feature_names()
decision_tree= random_forest.estimators_[random.randrange(0, 50)]
tree_rules = export_text(decision_tree, feature_names=feature_names)
print(tree_rules)
|--- ios13 <= 0.08
|   |--- iphone 11 <= 0.03
|   |   |--- after upgrade <= 0.03
|   |   |   |--- installing ios <= 0.03
|   |   |   |   |--- since updated <= 0.05
|   |   |   |   |   |--- xr running <= 0.05
|   |   |   |   |   |   |--- please fix <= 0.04
|   |   |   |   |   |   |   |--- not working <= 0.06
|   |   |   |   |   |   |   |   |--- 13 keyboard <= 0.05
|   |   |   |   |   |   |   |   |   |--- recent os <= 0.05
|   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 101
|   |   |   |   |   |   |   |   |   |--- recent os >  0.05
|   |   |   |   |   |   |   |   |   |   |--- class: 1.0
|   |   |   |   |   |   |   |   |--- 13 keyboard >  0.05
|   |   |   |   |   |   |   |   |   |--- class: 1.0
|   |   |   |   |   |   |   |--- not working >  0.06
|   |   |   |   |   |   |   |   |--- class: 1.0
|   |   |   |   |   |   |--- please fix >  0.04
|   |   |   |   |   |   |   |--- bug <= 0.06
```

Figure 6: Extract from decision tree of random forest classifier

An extract from a randomly chosen decision tree is illustrated as console output in Figure 6 above. The depicted decision tree incorporates word features along with conditional control statements. At each node of the tree, the TF-IDF weight of a feature is visible. Some tree rules were connected to the newest iPhone 11 or XR versions, others referred to events after the iOS update. Several different decision trees were scrutinized in this way, which unveiled further evidence considering the classifier's goodness.

In sum, the performance metrics and the analysis of categorized reviews provided important information for the model selection. As has been pointed out, the random forest classifier ended up being the best model. This outcome was in line with past studies which benchmarked random forests against other algorithms for text classification, whereby the superior discriminative power of random forests stood out (Fernández-Delgado et al. 2014, p. 3133). A second argument for selecting the random forest classifier was its transparent learning mechanism. Visualized tree rules made the decision-making process comprehensible. Finally, the ensemble method random forests showed no critical signs of overfitting. Its generalization capability enabled the identification of additional reviews with indirect iOS 13 reference. All model predictions were saved for the succeeding validation check.

5.2 Statistical summary of classified reviews

Descriptive statistics were fundamental in order to gain an in-depth understanding of the acquired feedback data. Feedback about iOS 13 was obtained through the two-step review classification process. This categorizing operation, which involved regular expression text search and random forests as machine learning algorithm, took into account both explicit and implicit mentions of iOS 13. It resulted in two different data sets that needed to be checked for conformity. For this purpose, the table below lists comparative key figures.

	iOS 13 reviews labeled via regular expressions	**iOS 13 reviews labeled via machine learning**	**Total data set**
Number of reviews	3,416	714	646,580
Average rating	2.38	2.44	3.54
Standard deviation of rating	1.45	1.48	1.71
Average characters per review	261	239	178
Standard deviation of characters per review	262	206	222

Table 5: Descriptive statistics of classified reviews

From 4,130 identified iOS 13 reviews, 3,416 were tagged via regular expressions and further 714 via machine learning. According to Table 5, the discrepancy in average review rating between the iOS 13 subsets amounted to merely 0.06 rating points, which was a relatively small difference in view of the rating standard deviations. The difference between the rating standard deviations themselves was just half as large with 0.03 points. Likewise, the figures relating to review length indicated similar properties of the classified subsets. In consequence of the evident conformity, all 4,130 records stemming from both classification approaches were considered as valid iOS 13 feedback. Besides, a comparison between iOS 13 reviews and the full data set evinced substantial disparities. Not only was the mean of iOS 13 review ratings over 1 point lower than the total average rating, but also the variability of the former was less. Concerning characters per review, labeled comments referencing Apple's thirteenth operating system update tended to be longer than the overall average. Yet, classified iOS 13 records constituted a rather small part of the whole data set, wherefore its concrete influences ran into danger of getting lost in the masses of other reviews. It was thus necessary to determine application subgroups with stronger repercussions of the iOS patch.

	Country		**Popularity rank**				
	GB	**US**	**1-2**	**3-4**	**5-6**	**7-8**	**9-10**
Number of iOS 13 reviews	427	3,703	1,939	819	357	411	604
Total number of reviews	63,352	583,228	285,105	114,241	55,061	84,950	107,223

Table 6: Breakdown of reviews across country and ranking dimensions

Table 6 describes the portion of reviews about iOS 13 for the dimensions of country and app ranking. The table values inside the country columns show that the majority of iOS comments were scraped from the United States App Store. Even though fewer records came from

Apple's British App Store, the relative shares of iOS 13 records for both marketplaces were roughly the same, namely 0.67% for Great Britain and 0.63% for the United States. These percentages signaled that, at large, the operating system update affected the two countries to the same extent. Another dimension within Table 6 is the ranking position. Applications ranked first or second had the highest and those ranked fifth or sixth had the lowest number of iOS 13 reviews. Their relative iOS 13 comments share was, however, very similar with 0.68% for the former and 0.65% for the latter ranking range. While the ranking range 3-4 had the highest ratio of iOS 13 comments, the positions 7-8 had the lowest, but the ratio increased again for the last two ranking places. Taken together, the ranking values gave no hint of an apparent relationship between app popularity and the ratio of iOS 13 references. Since neither country nor app ranking were clearly interconnected with the relative proportion of iOS feedback, app category was explored as a third dimension.

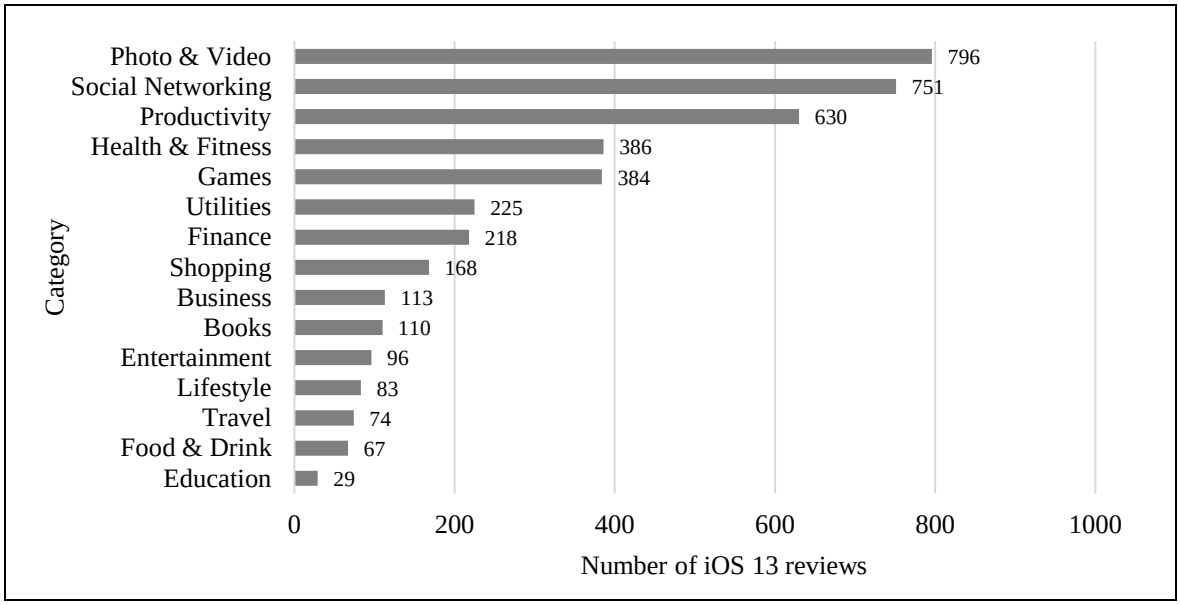


Figure 7: Distribution of iOS 13 reviews across app categories

The bar chart in Figure 7 displays the absolute number of iOS 13 reviews across all studied app categories, where each bar of the 15 categories aggregates records of the top ten selected apps in that group. It is visible that every single category contained user remarks about Apple's recent iOS version, which confirmed the update's broad impact. At the same time, by looking at the bar sizes above, an extremely uneven spread of user reviews was discernible between top and bottom categories of the chart. Putting it precisely, the top three categories made up more than half of the iOS 13 records with a 52.71% share, whereas the three smallest categories Travel, Food & Drink, and Education only contributed 4.12%. Aside from this observation, it is noteworthy that the total sum of user comments in a category was not proportional to the ratio of iOS 13 references. For instance, the categories Photo & Video, Games, and Entertainment held the largest portion of overall reviews, but as seen in the chart, they did not encompass the most comments about the iOS patch. The category Productivity

had the largest relative proportion of iOS 13 records with 5.25%, though it was the category with the second lowest number of comments among all others. Photo & Video as category with the most reviews, on the other hand, distinguished itself with a quite low ratio of 0.54% iOS 13-related comments. Therefore, there was no definite interrelation between the sheer number of iOS 13 references in a category and the actual relative distribution. As an intermediate result, none of category, country, or ranking dimensions could predicted the relative frequency of iOS 13 problems. The update repercussions were determined by a complex of multiple variables as opposed to a set of high-dimensional general rules. Consequently, the further analysis focused on individual applications which were especially affected.

App	Publisher	Category	Total number of reviews	Percentage of iOS 13 reviews
Gmail	Google LLC	Productivity	2,647	13.15%
Fitbit	Fitbit, Inc.	Health & Fitness	5,114	6.77%
Duo Mobile	Duo Security	Business	180	6.67%
Google Chrome	Google LLC	Utilities	701	6.28%
Amazon Flex	AMZN Mobile LLC	Business	979	6.03%
Google	Google LLC	Utilities	1,002	5.29%
Yahoo Mail	Yahoo	Productivity	2,052	5.21%
Microsoft Outlook	Microsoft Corp.	Productivity	2,133	5.20%
Dropbox	Dropbox	Productivity	310	5.16%
WhatsApp Messenger	WhatsApp Inc.	Social Networking	6,313	4.89%
Adobe Acrobat Reader	Adobe Inc.	Business	300	4.00%
Verizon Call Filter	Verizon Wireless	Utilities	386	3.37%
American Airlines	American Airlines	Travel	457	2.63%
Ring	Ring.com	Utilities	1,433	2.37%
Amazon	AMZN Mobile LLC	Shopping	4,828	2.30%
Google Home	Google LLC	Lifestyle	1,018	2.26%
Facebook	Facebook, Inc.	Social Networking	11,777	2.05%
Google Sheets	Google LLC	Productivity	371	1.89%
Audible	Audible, Inc.	Books	3,967	1.74%
Messenger	Facebook, Inc.	Social Networking	4,845	1.69%

Table 7: Top 20 apps with highest percentage of iOS 13 reviews

In Table 7, the 20 applications with the greatest relative share of iOS 13 comments are itemized. Instead of sorting by absolute comments, percentages were preferred to avoid clouding real update influences through potentially low ratios of iOS-specific content. Within the third column, 9 of the 15 distinct categories are represented. As has been mentioned before, Productivity had on average the highest percentage values and it is also the most prevalent category inside the table. Out of 150 sampled apps, an overwhelming majority of 119 comprised iOS 13 feedback. Among the 20 most affected products, the table includes widely popular apps, such as WhatsApp, Facebook, Amazon, Microsoft Outlook, or various products from Google. All these apps belong to professional developer studios and usually take up upper ranking positions. So, even product portfolios of reputable technology companies were subject to complications regarding Apple's newest operating system. During the sta-

tistical analysis, it was already noted that reviews with iOS 13 remarks tended to have substantially lower ratings than the total average. Following up on this, the ensuing step examined changes in sentiment as well as rating scores from applications impacted by the update.

5.3 Rating and sentiment analysis

In order to understand how App Store customers reacted after the release of iOS 13, the star rating system played a central role. Every scraped review was interlinked with such a rating ranging from 1 to 5 stars, enabling customers to communicate their overall evaluation in a single number. The numerical rating data allowed the usage of quantitative analyses. Moreover, it provided an excellent proxy measure for changes in users' continuance intention.

Complementary to app ratings, the sentiment of each review text was determined in the course of the data mining process. Depending on the positive or negative connotation of words present in those reviews, the sentiment analysis tool VADER calculated a positive polarity score between 0 and 1, with 1 being maximum positivity. It is worth mentioning that sentiment was not inevitably consonant with rating. The data set analysis revealed that, in a few cases, users complained about app problems after the iOS update, but they still liked the app and did not project these issues on the app itself giving it a 5-star rating. Nevertheless, rating and sentiment of a review were typically in accordance with another. As a first plausibility test of derived sentiment scores, a simple correlation analysis was conducted.

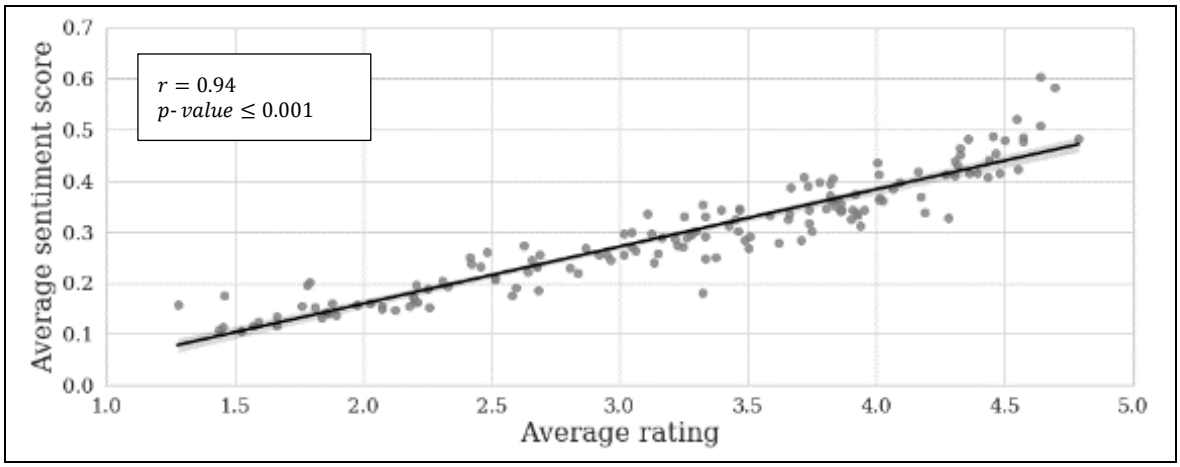


Figure 8: Linear relationship between rating and sentiment scores

The scatter plot in Figure 8 depicts rating against sentiment scores. Both average values were calculated and plotted for all 150 applications, which are marked as gray data points. On a closer look, there is a strong positive linear relationship because of the relatively small spread of points around the black regression line. Both variables move in tandem so that higher ratings go together with more positive sentiment and vice versa. To exactly measure the linear association between the two variables, Pearson's correlation coefficient, sometimes

referred to as Pearson's r, was used. It equals the covariance of the two variables divided by their respective standard deviations. Its values can vary from -1 to 1, where a value of 0 means that no linear relation exists and a value of -1 or 1 reflects a perfect negative or positive linear association, respectively. The computation of Pearson's r and its corresponding p-value relied on SciPy, a Python-based scientific computing library providing a large number of statistical functions (Virtanen et al. 2020, p. 1). With a correlation coefficient of 0.94, the very strong, positive correlation visible in Figure 8 was confirmed. A p-value ≤ 0.001 indicated an exceptionally low probability of obtaining a correlation coefficient from an uncorrelated data source that is at least as extreme as the one computed.

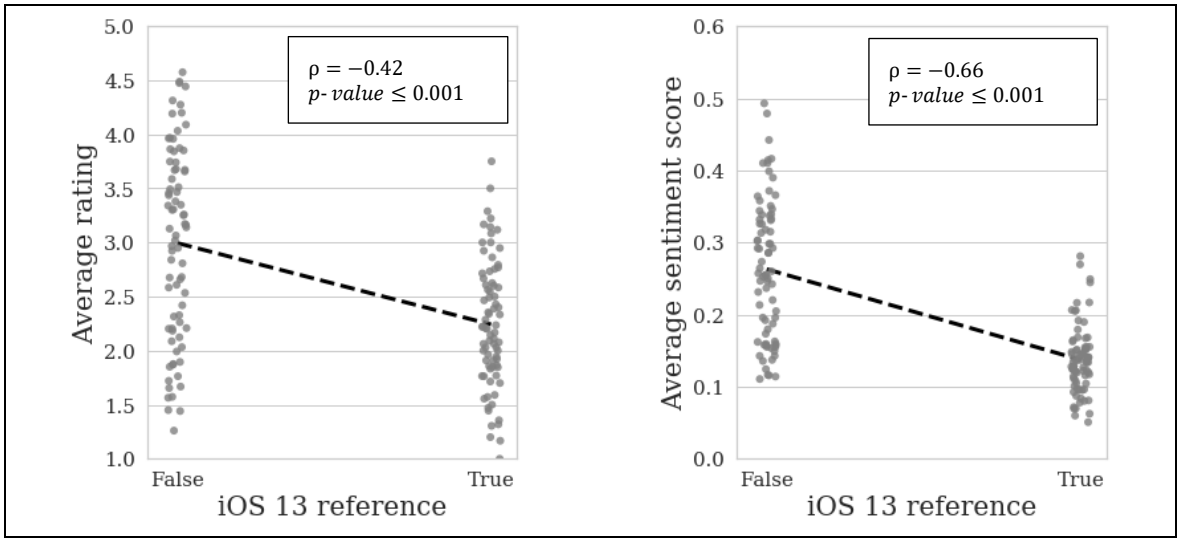


Figure 9: Rank-biserial correlation analysis results

A second correlation analysis concentrated on whether customer reviews that mention Apple's iOS update were associated with declining sentiment and rating scores. Similar to the preceding calculations, averages for rating and sentiment were computed, although applications with less than five references about the iOS patch were excluded. In addition, the mean values were calculated twice, once for iOS 13 and once for non-iOS 13 classified reviews. The results are illustrated in Figure 9 above. Data points mark average values for each of the 78 apps. The dashed line passes through the group means of the separated data point classes. According to Figure 9, average app ratings with iOS 13 reference turned out to be worse than ratings without. The latter had also a higher variation than the former. These findings were even more pronounced in respect of sentiment scores. While average positivity values for reviews without iOS 13 reference were scattered from 0.1 up to 0.5, app reviews about iOS 13 were clustered around 0.15 and below. Thus, upon visual inspection only, a negative relationship between user remarks on the iOS update and average app rating as well as sentiment scores was apparent. Furthermore, rank-biserial correlation using Spearman's ρ was consulted as supplementary evidence. It is the non-parametric counterpart of Pearson's r in situations when one variable is dichotomous. Unlike Pearson correlation, Spearman's rank

correlation does not assume that the data are normally distributed or homoscedastic (Spearman 1904, p. 80). Considering average app rating and iOS 13 reference, the correlation coefficient of -0.42 suggested a moderate negative correlation. As noticeable in Figure 9, the monotonic, negative correlation for sentiment was even stronger with a value of -0.66. The two estimated p-values attested that both calculated coefficients were highly significant.

	Month before release	First month after release	Second month after release	Third month after release	Grand total
Number of iOS 13 reviews	98	2,593	1,052	387	4,130
Average rating	2.49	2.48	2.23	2.19	2.39
Standard deviation of rating	1.55	1.49	1.38	1.36	1.46
Average sentiment score	0.14	0.15	0.13	0.12	0.15
Standard deviation of sentiment score	0.12	0.14	0.13	0.12	0.14

Table 8: Temporal changes in rating and sentiment scores of iOS 13 reviews

In a subsequent analysis step, temporal rating and sentiment changes in iOS 13 feedback were explored. Table 8 includes relevant numbers over the four-month observation period from 19 August till 19 December 2019. During the month before its release date on 19 September, the iOS 13 update was cited in merely 98 reviews by users with access to the public beta version. Customers wrote the bulk of iOS 13 reviews within the first month after release with a total of 2,593 records, whereby this number was more than cut in half in the second and again in the third observation month. Besides, the table shows a substantial drop in average rating between the first month and the following two months after the iOS release. The same applied for average sentiment scores, albeit in weakened form. Standard deviations for both rating and sentiment were lowest during the last two observed months. Put concisely, iOS 13 feedback became less prevalent over time, but its negative effect on app rating intensified along with a reduction in positive review sentiment.

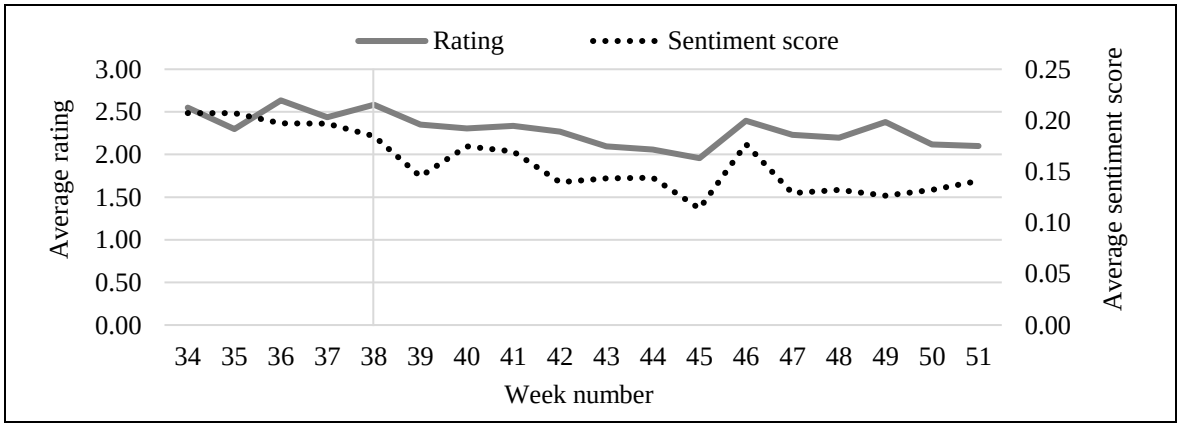


Figure 10: Weekly rating and sentiment scores of Gmail app

By means of an example, weekly rating and sentiment values for Google's Gmail app, which had the highest relative share of iOS 13 references, are displayed in Figure 10 above. The

data basis consisted of 2,647 scraped Gmail reviews. Calendar weeks are listed on the abscissa. A vertical line at week number 38 marks the week of the update release. As depicted in the graph, both average rating and positivity score started to deteriorate after week 38. The decline continued until a trough was reached in week 45. It was not solely caused by the 348 classified iOS references, but also by hitherto undetected iOS 13 feedback. The decrease in rating and positivity occurred mainly throughout the first and second month after the iOS patch. Then, the values slightly recovered within the remaining weeks. Across the entire observation period, sentiment and rating were correlative in their weekly changes. WhatsApp, Yahoo Mail, or other apps which were heavily impacted by iOS 13 exhibited a similar development like in the case of Gmail. Altogether, the rating and sentiment analysis of individual apps validated the findings that were discovered to this point.

5.4 Evaluation of topic model and update issue types

The LDA topic modeling algorithm was employed for discovering thematic dimensions contained in a collection of app reviews. Aside from review documents, LDA required a predefined number of topics as an input parameter. To ascertain the optimal number of topics, multiple LDA models with varying topic numbers were trained, compared and graphically visualized. The two curves in Figure 11 denote changes in coherence and perplexity scores for models with different topic numbers ranging from 2 to 20. In terms of coherency, starting out from two topics, coherence scores improved with an increasing number of topics, peaked at six topics and then gradually deteriorated for larger numbers. Due to the innate variability of LDA models, which originates from the initial random topic assignments, several training repetitions were executed to verify this curve progression. In consideration of model perplexity, the calculated negative logarithmic perplexity decreased with the number of topics, which is associated with better generalization performance (Blei et al. 2003, p. 1008). Regarding the intention of identifying consistent update issue themes, coherency was a more decisive factor for parameter optimization, wherefore a topic number of six was chosen.

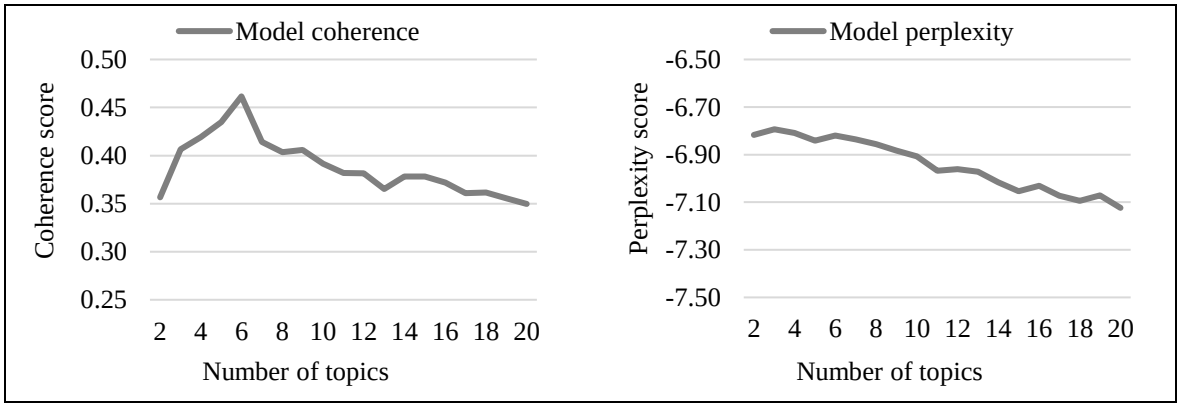


Figure 11: Topic model coherence and perplexity

In the ensuing analysis step, the modeled topics were interpreted. A common practice for making sense of topic models is to look at the most relevant words within each topic. To this end, a term relevance metric was used from LDAvis, a web-based interactive visualization of LDA generated topics (Sievert and Shirley 2014, p. 63). Below, Table 9 provides the top 15 most relevant terms for each of the six modeled topics.

Topic	Top 15 topic-related terms sorted by relevance metric[7]
1	[“dark mode”, “add”, “support”, “need”, “dark”, “theme”, “instagram”, “please”, “night”, “option”, “feature”, “eye”, “white”, “facebook”, “no”]
2	[“crash”, “bug”, “ios”, “app”, “open”, “iphone”, “audio”, “freeze”, “keyboard”, “close”, “13.1.2”, “running”, “pro”, “file”, “email”]
3	[“account”, “keep”, “log”, “app”, “widget”, “balance”, “connection”, “logged”, “bar”, “see”, “safari”, “constantly”, “crashing”, “sign”, “amazon”]
4	[“notification”, “watch”, “fitbit”, “text”, “sync”, “versa”, “bluetooth”, “stopped”, “audible”, “no longer”, “sound”, “charge”, “receive”, “syncing”, “beta”]
5	[“search”, “don’t”, “issue”, “time”, “download”, “sleep”, “week”, “item”, “get”, “font”, “show”, “service”, “ad”, “else”, “answer”]
6	[“game”, “controller”, “controller support”, “control”, “ps4”, “xbox”, “multiplayer”, “mobile”, “fun”, “cod”, “graphic”, “played”, “race”, “ruby”, “duty”]

Table 9: Top 15 most relevant terms per modeled topic

Looking at the table, the first topic included the bigram “dark mode” as most relevant term as well as analogous terms like “night”, “option”, “dark”, or “theme”. Its thematic direction was straightforward. There was an obvious reference to the newly introduced setting in iOS 13 that darkens user interface elements. The words “add”, “feature”, and “please” pointed to lacking support of the dark iOS theme in third-party applications. Instagram and Facebook appeared, among others, in the most relevant words as apps without dark mode support.

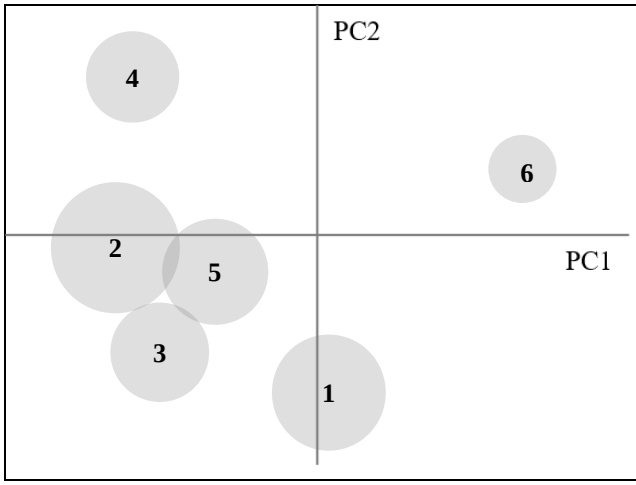
The second and third topic were contextually related, since both encompassed relevant words about software errors, such as “bug”, “constantly”, “crashing”, or “freeze”. While the former tended to address malfunctions in respect of audio, keyboard, and file management, the latter focused on log-in and app widget issues. Screening the review texts, concrete cases were identified, for instance, the Gmail app crashed when trying to open audio file attachments and the Capital One app had a broken account balance widget. As a matter of fact, the fifth topic was thematically related to software errors, too. For example, it involved the terms “search”, “issue”, and “font”. These terms reflected problems across several applications concerning a defect search function or misrepresented font formatting.

The fourth topic principally evolved around the context of smart devices. There were thematic connections to Apple’s smartwatch and activity trackers from the company Fitbit. Some of its topic-related terms like “notification”, “bluetooth”, “stopped”, or “syncing” evinced the presence of device communication failures. In fact, the vast majority of reviews under this topic mentioned that, after installing the iOS 13 update, the Fitbit device stopped

[7] Relevance(term w|topic t) = $\lambda * p(w|t) + (1 - \lambda) * p(w|t)/p(w)$ with $\lambda = 0.4$ (Sievert and Shirley 2014, p. 66)

receiving text notifications from the iPhone and, in addition, a significantly increased battery drain was reported. In regard to the Apple Watch, users of Audible, Duo Mobile, and other apps reported that their smartwatch refused data synchronization leading to severe incompatibility issues with the applications in question.

Lastly, the sixth topic had a clear link to mobile games. Out of 244 reviews under this topic, 225 feedbacks came from the popular gaming apps Call of Duty Mobile and Mario Kart Tour. The bigram "controller support" as well as the words "ps4", "xbox", or "multiplayer" were amongst the most relevant terms. They referred to a new feature of Apple's iOS update that allowed customers to connect a wireless controller to their iPhone via Bluetooth. Some apps, like the two mentioned above, did not support controller pairing yet. Hundreds of users expressed criticism over this situation in their app reviews.



Issue type	Topic	Percentage of tokens
Feature request	1	20.40%
	6	6.90%
Functional error	2	26.10%
	3	15.30%
	5	18.30%
Device compatibility	4	13.00%

Figure 12: Intertopic distance map and topic grouping

A further step towards discovering specific issue types caused by iOS 13 was to find semantic similarities between the modeled topics. For this purpose, the visualization framework LDAvis was leveraged to project the intertopic distances into a two-dimensional space with principal component analysis (PCA). To put it short, PCA applies orthogonal transformation to convert a multidimensional feature space into a set of linearly uncorrelated variables called principal components, which account for the most variance of the data. As displayed in Figure 12, a topic is plotted as a circle whose size is proportional to its relative share of word tokens in the corpus. The minimal overlap between the circles indicated a good model quality, though, it was apparent that the second, third, and fifth topic slightly overlapped and clustered together. Because all of these three topics contained reviews about crashing applications or dysfunctional features, they were grouped under the issue type of functional errors. Besides, the intertopic distance map illustrates that the first and sixth topic are positioned closest to the right-hand-side of the PC1 axis. As another common characteristic, the two topics were not centered around erroneous software, but rather bundled user reviews which

articulated the need for supporting new features introduced in iOS 13. Therefore, both topics were allocated to the feature request issue type. In the upper left quadrant, the fourth topic is positioned above the functional error topic cluster. Despite of its semantic resemblance to software failures, the fourth topic concentrated specifically on malfunctions of smart devices. Bearing this differentiating aspect in mind, it was categorized under a third issue type named device compatibility. Eventually, three distinct issue types were established from the six modeled topics. Owing to LDA's soft clustering, each document had a probability distribution over these six topics instead of one discrete topic, which is why the topic with the highest likelihood was assigned to each document of the corpus. When counting all documents in the topic grouping, the three types functional error, device compatibility, and feature request consisted of 2,229, 543, and 1,358 user feedbacks, respectively.

In the course of the preceding topic model evaluation, review texts were scoured for widespread issues that manifested after Apple's latest iOS patch. Three very distinct update issue types emerged during this process. Structure and content of the reported problems varied greatly depending on the issue type. In a final step, it was tested if these substantial differences were likewise mirrored in rating and sentiment scores.

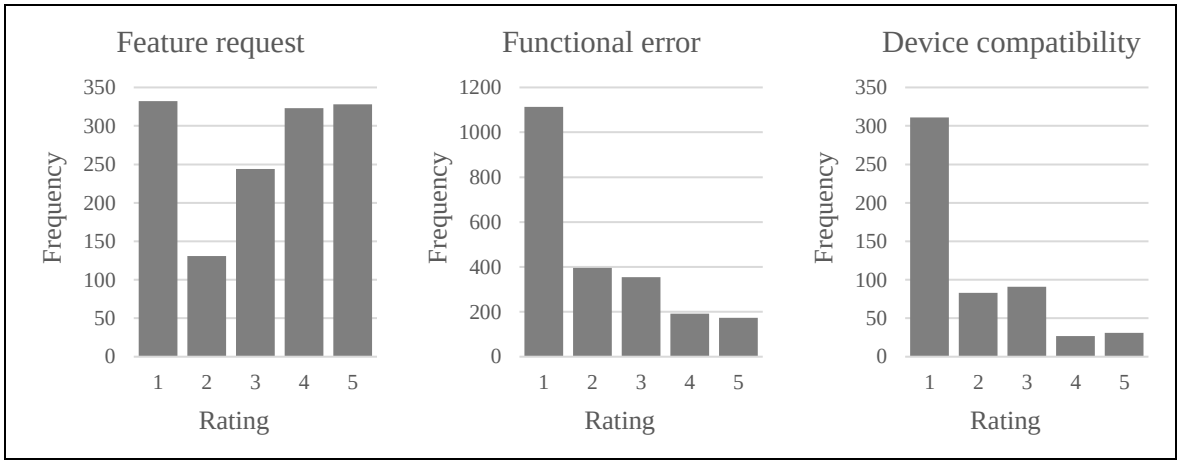


Figure 13: Rating histograms for update issue types

An initial overview of the frequency distribution of ratings for each issue type is given in Figure 13. What is striking is the resemblance of distributed ratings for functional error and device compatibility. In Figure 13, they are both characterized through a similar unimodal shape. It is recognizable that both distributions are highly right-skewed. Moreover, they both had a mode of 1, or in other words, the most frequent user feedback was a one-star rating. Apart from this, their median rating was not equal with a value of 2 in the functional error and 1 in the device compatibility histogram. Also, 4-star and 5-star ratings were the least frequent records. By contrast, the feature request histogram had a very different underlying distribution. Rating scores of this issue type approximated the shape of a bimodal distribution. There were nearly as many 1-star as 5-star app ratings, followed by 4-star ratings.

Thereby, the standard deviation of the rating was much larger than that of the device compatibility or functional error distributions. The median rating of 3 was higher than in the other two update issue groups. In accordance with ratings, the sentiment score histograms showed analogous dissimilarities in skewness and variance.

With the help of statistical methods, the impact of the three issue types on user feedback was scrutinized in depth. One-way analysis of variance (ANOVA) is a technique for determining the existence of statistically significant differences among at least two grouped samples. This is tested by comparing the between versus the within group variance. However, the acquired data did not satisfy the restrictive ANOVA assumptions, such as homogeneity of variances and normally distributed observations. This was already outlined in the passage above. For one thing, rating and sentiment score had either a bimodal or right-skewed distribution, and for another, the standard deviations of the grouped samples were different.

The Kruskal-Wallis test, also called *H*-test, is a non-parametric analog to the one-way ANOVA, which means it neither makes assumptions about normality nor homoscedasticity (Kruskal and Wallis 1952, p. 583). A significant *H*-test statistic result signals the stochastic dominance of at least one grouped sample over any other group, or otherwise stated, it is more likely that a randomly drawn observation from one sample will be greater than a random observation from another. But as an omnibus test, it does not pinpoint where this dominance occurs. Thus, upon rejecting the null hypothesis, post hoc pairwise comparisons are conducted if necessary. Independence of the samples was guaranteed by the deletion of reviews from authors who were present in more than one issue group. The computation of the *H*-statistic and p-value relied on Scipy's Kruskal-Wallis implementation.

Descriptive statistics			
Issue types	**Number of reviews**	**Median rating**	**Median sentiment score**
Feature request	1,358	3	0.23
Functional error	2,229	2	0.10
Device compatibility	543	1	0.08
Test statistics			
Tested issue types	**Degrees of freedom**	**Rating**	**Sentiment score**
Feature request Functional error Device compatibility	2	*H*-statistic = 531.53 p-value ≤ 0.001	*H*-statistic = 533.29 p-value ≤ 0.001
Functional error Device compatibility	1	*H*-statistic = 8.57 p-value ≤ 0.01	*H*-statistic = 6.38 p-value ≤ 0.05

Table 10: Kruskal-Wallis test results

Table 10 summarizes the calculated test results. Considering user ratings across all three issue types, the *H*-statistic, along with its highly significant p-value, indicated the existence of stochastic dominance. The same held true for sentiment scores. Looking back at the histogram analysis, signs of dominance of the feature request group were foreseeable, since its

rating and sentiment distributions were less skewed towards lower scores than the distributions of the remaining issue groups. To compare the latter two groups in detail, a post hoc test between the functional error and device compatibility issue type was done. As could be seen from the histograms, both sample distributions had similar shapes and variances for rating and sentiment scores. In such a special case, the Kruskal-Wallis null hypothesis is interpreted as a statement about the equality of the group medians, whereas the alternative hypothesis proposes the opposite. According to the figures in Table 10, the very significant *H*-statistic of 8.57 suggested that the median ratings of the functional error and device compatibility samples were unequal. In particular, app ratings of reviews concerning device compatibility issues proved to be more negative than of reviews about functional errors. The same applied to sentiment scores, even though with a smaller *H*-statistic of 6.38. Still, the corresponding p-value turned out to be significant, wherefore the null hypothesis, which assumed equal group medians, was rejected. Once again, the findings for rating scores were consistent with those of sentimental positivity. Through the Kruskal-Wallis test, substantial differences between each one of the three issue types were demonstrated.

6 Discussion

The following chapter elaborates upon the results of the empirical analysis from chapter 5. Relevant findings are reviewed and interpreted with respect to the thesis statement. In this way, the three formulated research hypotheses are examined in consecutive order by critically discussing the empirical evidence.

The first hypothesis (H1) assumed that the release of the iOS 13 patch negatively affected app users' continuance intention. Consequently, it was essential to detect potential negative feedback about the recent iOS version in customer reviews. A two-step classification process labeled review texts with explicit or implicit remarks on iOS 13. Because filtering out implicit references happened to be a non-trivial task, naïve Bayes, logistic regression, and random forest classifiers were benchmarked against each other. Owing to its superior performance, including a F_1-score of 0.96, the latter model was selected. This choice can be explained by the inherent benefits of random forests. Especially worth mentioning are the abilities to deal with high-dimensional text data or to handle unbalanced classed distributions, which were very advantageous in respect of the given review data set. The extracted tree in Figure 6, offers insights into the underlying decision rules. It seems reasonable that users implicitly reference iOS 13 by naming their affected iPhone generation, pointing out specific bugs, or referring to the time period after the iOS update. This underlines the good reliability of the trained random forest model. Additionally, the outcome of the model comparison is in line with machine learning literature, for instance, in an exhaustive evaluation of 179 classifiers from FERNÁNDEZ-DELGADO et al., the best performance was achieved by the family of random forest algorithms (Fernández-Delgado et al. 2014, p. 3175).

In spite of the model's effectiveness, a relatively small number of 4,130 reviews were found to have a relation to iOS 13, which falls behind initial expectations. This can be attributed to two factors. On the one hand, mobile app users diverge in their individual technology expertise (Fleischmann et al. 2015, p. 5). While experts have more knowledge to assess the root causes behind app issues, novice users may not associate malfunctions with an iOS update, not to speak of knowing the version number of their currently installed operating system. Therefore, a large number of comments from novice users is probably lacking any clues to iOS 13, even if they experienced problems with it. On the other hand, the strictness of the classification model could imply that some feedbacks with implicit, superficial iOS 13 remarks were missed, yet, the model's stringency is a desired property. This way, the classification results were accurate, and the analyses were not impeded by too many false positives.

With the hypothesis H1 in mind, rating and sentiment scores of the classified reviews were examined to gauge users' continuance intention. The figures in Table 5 unveil that the average rating of iOS 13 reviews is worse than the total average rating. Likewise, user comments

related to iOS 13 have on average more characters per review. One plausible explanation for this phenomenon is the tendency of customers to put more effort in their feedback when they are dissatisfied with a product or service. They often describe negative experiences in great detail. Typically, more words are used to express anger, disappointment, or impatience. This argumentation is backed up by previous research studies about electronic word-of-mouth feedback (Chevalier and Mayzlin 2006, p. 345; Verhagen et al. 2013, p. 1430).

Through determining review sentiment in the form of positivity scores, a complementary measure for assessing the intention of sustained app usage was attained. The observed strong relationship of rating and positivity is considered as validation of the calculated scores. It appears rational to expect that, in general, better ratings go together with more positive user comments. Otherwise, the reviews would be contradictory and difficult to interpret. The outcome of the correlation analysis in Figure 9 shows a highly significant negative correlation between iOS update-related feedback and rating as well as positivity scores, with coefficients of -0.42 and -0.66, respectively. In accordance with these observations, the examination of Google's Gmail app, which had a particularly high ratio of iOS 13 comments, reveals a visible drop of rating and sentiment scores occurring after the iOS release date, succeeded by moderate improvements during November and December. Although it is believed that the scores will further recover due to the debugging endeavors of developer studios. Naturally, more long-term data is necessary to support this assumption, whereas the present study focused on the immediate update effects. Following the ISCM logic, it can be thus argued that Apple's new iOS version caused lots of unexpected application errors. Since those issues heavily impair the usage experience, it is presumed that customers are negatively disconfirmed. This also finds expression in dissatisfaction or in a worsened perception of an app's usefulness, as has been proven by the analyzed decline in rating and sentiment scores. Ultimately, diminished levels of satisfaction and perceived usefulness are believed to have led to a reduction of users' continuance intention (Bhattacherjee 2001, p. 351). To conclude, the discussed findings provide enough empirical data to accept hypothesis H1.

The second hypothesis (H2) suspected that discernible types of application issues resulting from the iOS 13 update are manifested in user reviews. In view of the collected feedback, there was a sheer endless number of reported bugs, crashes, feature requests, network issues, performance losses, and other complaints. As an attempt to arrange individual problems into a typology of update-related issues, the topic modeling algorithm LDA was employed. LDA assisted in inferring latent themes within the corpus of review documents. The graph in Figure 11 illustrates how the model coherence score changes with the topic number. It is not surprising that coherency peaks at a number of six topics. A model with merely two or three topics may be unable to differentiate unique themes, thereby failing to uncover the true underlying semantic structure. On the opposite side, a model with too many topics may produce

ambiguous, overlapping topics, which can complicate the topic assignment. Another possible hindrance for allocating documents to one out of several themes concerns the review content itself. Scanning through the collected comments, it becomes apparent that some users raise not one, but multiple app issues in their commentary, as has been also noted in related research (McIlroy et al. 2016, p. 1067). To address this problem, the six modeled topics were categorized into three well-separated issue types.

Functional errors constituted a first type, which accounted for over half of the classified feedback records. It grouped together manifold software problems, ranging from minor bugs to permanent app crashes. In order to avoid semantic overlaps, it seems appropriate to merge all kinds of functionality issues into one type because they tend to frequently co-occur in user reviews. For instance, several App Store customers wrote about various dysfunctions and warned others to not install the new iOS patch: “I am relegated to deleting the app entirely and going browser-based. It is chock full of bugs and does not work well with iOS13. Do not update to the latest version. The lag now experienced while typing and general navigation is unbearable.” Obviously, update-induced functional impairments are widely represented in the collection of review texts. The problem descriptions inside those user reviews are believed to be a crucial information source for monitoring and fixing app errors.

Device compatibility emerged as a second issue type. Despite of the semantic resemblance to the functional error type, it can be argued that incompatibility issues belong to a special quality of iOS 13 repercussions, in which not only iPhone applications, but also peripheral devices are negatively impacted. The content analysis exposed common troubles with Apple’s own smartwatch series or third-party products like activity trackers from Fitbit. In numerous comments, users vehemently complained about malfunctioning devices after they switched to the new operating system: “Ever since updating to 13.1.3 I have stopped receiving text message notifications to my Fitbit device. I have contacted Apple and Fitbit support to no avail. Fitbit has been aware for a long time of the issues that many customers are experiencing with iOS 13 and have yet to address the issue.” From the reported compatibility problems, a broken data transfer between iPhones and smart devices appears to be very prevalent. Without this constant synchronization, it is likely that many features of an affected device become useless, much to the annoyance of customers who bought the device.

Feature requests were the last identified issue type. In contrast to the subgroups above, this type is not linked to arisen defects. Instead, it encompassed uttered user wishes to add novel functions. In fact, the requests revolved around newly introduced features of the iOS 13 update, such as the option to connect a Bluetooth controller or a novel dark color scheme setting. It is comprehensible that users of the new operating system want to fully exploit its functionality. For this reason, it is astonishing that many developers, who normally have

access to beta versions of Apple's operating system, were not able to anticipate these user wishes. The number of applications with corresponding feature requests exceeds prior expectations. For example, a plethora of users asked for dark mode support and directly reached out to developers: "Can we finally have dark mode on all of your IOS apps please?!?!?! It was well known since JUNE that IOS 13 will have dark mode!" Besides, it is remarkable that even highly popular apps like Facebook, Instagram, Snapchat, WhatsApp, or Youtube failed to support Apple's dark mode from the beginning, though, for some of them, the feature was delivered subsequent to weeks of pressure from customers.

The above elaboration of problem categories connected to iOS 13 proves that it is feasible to extract knowledge about manifested issue types from user feedback. This is also in line with related studies which underpin the high informative value of mobile app user reviews (Chen et al. 2014, p. 767; Genc-Nayebi and Abran 2017, p. 207; Panichella et al. 2015, p. 281). On the basis of these results, hypothesis H2 is accepted.

The third hypothesis (H3) postulated that the influence of different issue types on app users' continuance intention varies in terms of magnitude. Its underlying reasoning expects a more negative effect of those problems which undermine the user experience to a greater extent. Against this background, rating and sentiment scores of the three established issue categories were analyzed in depth. Figure 13 gives an overview of how star ratings are distributed across the groups feature request, functional error, and device compatibility. As distinguished from the two latter ones, the feature request histogram is by far less skewed to the right, meaning it has a smaller relative share of low ratings. It has the highest median star rating among the groups and almost the same count of 1-star and 5-star evaluations. These figures endorse the initial assumption that users may perceive missing features as less critical than actual software flaws. A conceivable explanation for the still large number of one-star ratings concerns the inclination of individuals to pay more attention to negative events. This tendency to overweigh negative experiences, also referred to as negativity bias, has been established as a key principle in consumer psychology (Ahluwalia 2002, p. 270; Rozin and Royzman 2001, p. 296). Accordingly, it can be argued that app users who take note of an unsupported iOS feature tend to focus on its absence and are thus negatively disconfirmed. Owing to their dissatisfaction, they give lower ratings and eventually may have a weakened continuance intention, even though the application in question works just fine. This insight is believed to be vital for managing an app's feature set. For instance, a lot of users made their rating dependent on a requested functionality: "This will have a one-star rating until dark mode is a feature!!! iOS 13 has been out for many weeks now and still no dark mode."

Nonetheless, high ratings were yet more dominant in feature requests than in the remaining two types. The same applied to positive sentiment, as the highly significant Kruskal-Wallis

H-statistic of 533.29 indicated. In order to explicate why functional errors and device compatibility issues are perceived more negatively by users, the loss aversion principle from the domain of cognitive psychology can be used. Loss aversion describes the human tendency to judge losses as more weighty than corresponding gains (Kahneman and Tversky 1979, p. 263). If this rationale is transferred to the software update context it is assumed that, in general, users should prefer avoiding feature losses to acquiring equivalent new features, since the actual app usage is stronger affected by impairments of the existing functionality. Thinking one step further, it makes sense that the greater the scope of the loss, the more negative the effect. An erroneous app is often undeniably less severe than a malfunctioning device. In this regard, a pairwise comparison of median rating and sentiment scores for the groups functional error and device compatibility was conducted. The two significant *H*-statistics suggest lower median values for device compatibility on both accounts. Hence, it turns out that users perceive incompatibility of devices as most critical out of all issue types. High financial investments and switching costs may be exemplary reasons. There are plenty of reviews in which customers emphasized their strong discontent over impacted devices: “Text message notifications still do not work now three months after iOS 13 release. I will never buy another Fitbit product again. Do not recommend any of their products designed to receive text message notifications to anyone.” Without any doubt, the presence of device compatibility issues reinforces users’ discontinuance intention to a high degree.

In summary, the discussed findings help to differentiate the impact of the three issue types on rating and positivity scores. They proved that the influence on the intention of sustained app usage varied for each issue group. As a logical consequence, the results provide enough evidence to accept hypothesis H3.

7 Conclusion

In this concluding chapter, the main findings of the thesis at hand are briefly summarized with reference to the core research question. Thereafter, based on the obtained study results, theoretical as well as practical contributions are discussed. The final section points out the study's limitations and offers suggestions for future research.

7.1 Summary of key findings

The thesis at hand sought to study the influences of an operating system update on app users. The case of Apple's thirteenth update of the iOS platform served as object of investigation. On the basis of existing literature and the information systems continuance model as theoretical lens, three hypotheses were formulated about the adverse impact of iOS 13 on end users in Apple's mobile app ecosystem. In order to carry out the research, vast amounts of feedback data were scraped from the App Store. The collected data set comprised 150 sampled applications with a total of 646,580 reviews. Data mining techniques and statistical methods were used to test the proposed hypotheses, which resulted in several insightful findings regarding the observed influences of Apple's iOS update, thereby answering the research question. The key findings can be summarized as follows.

First, the empirical analysis demonstrated significant negative effects of iOS update-induced issues on users' satisfaction levels. The correlation analysis indicated that reviews with reference to iOS 13 were associated with lower ratings and less positive sentiment. The feedback also became more negative with each passing month after the release. Such critical remarks on the new iOS version were widespread in Apple's mobile app ecosystem. In fact, they could be diagnosed in the sample across all 15 product categories, across all ten popularity ranks, and in both the Great Britain and United States marketplaces. Furthermore, even reputable developer studios from major technology companies like Amazon, Facebook, Google, or Microsoft were not spared and had to resolve numerous complications with their products. The empirical data showed evidently that end users seemed to have very little tolerance for issues related to iOS 13. Consequently, there was no doubt that the adverse effects of the iOS update considerably reduced users' intention of continued app usage.

Second, the research investigated the manifestation of iOS platform issues in the acquired customer reviews from the App Store. A large number of discontented users wrote detailed descriptions about specific troubles they faced. The feedback texts proved to be a valuable data source, since a multitude of software complications could be identified. Through clustering the respective application flaws into separate categories, three overarching groups emerged, namely feature requests, functional errors, and device compatibility. For each of

these three issue types, a content analysis revealed insights into the expressed needs of customers. While some insisted on the implementation of iOS-specific features like dark mode or Bluetooth controller support, others reported a high incidence of broken functionalities, and yet others complained about malfunctioning devices since the update. The documented software problems ranged from occasional lags and minor bugs to complete failures. An impaired data synchronization between iPhones running iOS 13 and peripheral devices was ascertained as the primary cause for incompatible activity trackers, smartwatches, or media streaming hardware. Hence, the manifold technical influences of the iOS 13 update were noticeably reflected in feedback comments of the collected review data.

Third, the differentiated issue types were examined in terms of impact on users' continuance intention. To this end, rating and positivity of reviews within each group were compared against each other. According to the Kruskal-Wallis test results, the issue types exhibited significant differences. Device compatibility issues provoked the strongest negative reactions, followed by functional errors and feature requests. In the latter group, rating and sentiment scores were higher than in the two former groups, although many users penalized the lacking feature support with one-star ratings. Incompatible devices and dysfunctional apps attracted the most criticism of customers because compromised core functionalities ruined the general usage. Thus, it was found that the negative effect on users' continuance intention varied between the distinct types depending on the severity level of issues.

7.2 Theoretical and practical implications

The summarized findings illuminate how users perceive operating system updates and their effects on applications, framed in the context of mobile app ecosystems. In view of Apple's recent iOS 13 platform, an overwhelming body of evidence exemplifies the disruptions connected to its release. Having scrutinized its consequences in depth, this thesis provides both theoretical and practical contributions, which are set out in the section below.

From a theoretical standpoint, this thesis adds to the emerging research stream on software updates in the information systems post-adoption literature. To the author's best knowledge, it is the first to study how mobile app users perceive platform updates. Unlike existing work, it does not focus on the direct effects of intra-system updates, but on the indirect, interdependent update effects between operating system and application software. In accordance with relevant information systems literature, this thesis shows that software updates can have measurable effects on end users (Amirpur et al. 2015, p. 13; Claussen et al. 2013, p. 186; Fleischmann et al. 2016, p. 83; Hong et al. 2011, p. 235). Considering the situation of Apple's iOS 13 platform, the discoveries underline that its release was accompanied by a wide array of technical complications, much to the chagrin of App Store customers. In that respect,

the empirical analysis demonstrated that iOS-related issues impacted users' continuance intention in a negative way. This observation aligns with previous studies which corroborated adverse effects of certain software updates on continued information systems usage (Fleischmann et al. 2015, p. 17; Foerderer and Heinzl 2017, p. 15). For example, in line with prior evidence, it was confirmed that losses in functionality through an update severely diminishes the continuance intention of users. This can increase their willingness of switching to another product (Recker 2006, p. 21).

Adding to this observation, the results of the topic analysis proved that the severity of update-induced issues is closely tied to the negativity of user reactions. In this sense, the present study gives meaningful insights into how users perceive diverse issue types, such as lacking features, dysfunctional features, or incompatible devices. Accordingly, each of those issue types can have a different impact on the intention of sustained app usage. An additional noteworthy finding relates to feature requests arising due to the new iOS version. Customers turned out to be negatively disconfirmed if their installed apps did not support novel features of iOS 13 from the beginning, what means that an operating system update can alter customers' evaluation of initially accepted application software and may also reduce their continuance intention. Overall, the findings contribute to aid the current understanding of the complex set of influences that software updates can exert on end users.

Furthermore, the research results have important practical implications for actors in mobile app ecosystems. Taking iOS 13 as an exemplary case, the thesis shed light on potential adverse effects of platform updates. This is especially relevant for platform owners like Apple, as they must control the rapid evolution of platform technologies. The main challenge lies in continuously advancing the platform's capabilities through updates without, at the same time, introducing technical difficulties, which, in the worst scenario, can disturb the ecosystem and impede its members. The latter happened to be true for rolling out iOS 13. The empirical analysis gives a first impression of its far-reaching shortcomings that affected the vast majority of sampled apps. Owing to the central role of platforms, it is recommended to extensively test pre-release versions. However, in spite of Apple's beta software program for consumers and developers, the first iOS 13 version came out with a myriad of deficiencies. Therefore, a reexamination of software quality assurance and testing practices might be helpful. For counteracting problems after the official release, platform owners should work together with the end user as well as developer community. In close collaboration, they could engage to keep records of software flaws or incompatible devices. This could be made possible by creating an online network where users and developers jointly report on existing platform-related complications and so aid the dissemination of problem-specific knowledge.

Alongside recommendations for platform orchestrators, the thesis provides practical insights for app publishers. In particular, the manifestation of discernible updated-related issues in review texts was evinced. Developers are advised to leverage the valuable data contained in feedbacks in order to effectively fix application faults. They should also prioritize resolving certain issue types, as some issues were found to have a more destructive effect on sustained product usage, such as incompatible hardware or permanently crashing apps. Otherwise, the companies may run into danger of losing dissatisfied customers to the competition. The review analysis indeed found corresponding reviews about incompatible devices, in which customers uttered the intention of never buying a product from the company again or vehemently dissuaded other customers from doing so. In addition, the research results evidenced a substantial change in customer demands regarding their applications after the platform update, mainly because of newly introduced iOS features which were not yet supported. Hence, it is recommended that developer studios try to anticipate emerging customer requests. In this matter, an open information exchange between platform owner and third-party developers is indispensable to communicate the new platform functionalities and reconcile the feature sets of operating system and applications. This will not only increase user satisfaction, but also strengthen the customer base and, in turn, contribute to the general success of the ecosystem, which is a desirable outcome for all ecosystem members.

7.3 Limitations and future research

Despite the contributions to research as well as practice, the findings of this study are subject to several limitations. To start with, the results are limited by the data sample size. Due to constraints in terms of computing resources and time, data from 150 applications were collected, which is a small sample from the population of over two million available apps on the App Store. For this reason, the study does not claim to represent the full scope of application issues caused by iOS 13. For another thing, the observation time was restricted to a four-month period, since the study concentrated on the immediate effects of platform updates rather than the long-term consequences. The latter opens up a fertile avenue for possible future research. What is more, the text mining analysis was confined to comments in English language stemming from the American and British App Store. Therefore, cultural or regional dissimilarities might imply different customer perceptions. Even though it is probable to observe similar user reactions in other App Stores, cross-cultural studies are needed to verify this. The last limitation relates to the incomplete detection of feedback with iOS 13 reference. It is acknowledged that a plethora of users without technical background knowledge failed to diagnose the iOS update as error cause for their app problems. Their review texts possessed neither explicit nor implicit remarks on the operating system update and were thus not recognized during the classification. That is why the findings must be interpreted with careful attention to the existence of enormous amounts of latent feedback regarding iOS 13.

Aside from the stated limitations, the study results can be used as a starting point for further research. One option could be an extensive follow-up study with a larger data sample. Both free and paid applications in all listed categories from app marketplaces across all continents could be included for greater representative power. Additionally, a prolonged observation period may assist in investigating long-term update effects, such as the foreseeable recovery of rating scores. With this in mind, it seems promising to inspect the troubleshooting methods and the quickly deployed bug-fixing patches of Apple and developers as measures against the shortcomings of iOS 13. The effectiveness of these coping strategies could be assessed by analyzing changes in app user ratings over time. Apart from Apple's iOS platform, researchers are encouraged to study the influences of platform updates in other mobile app ecosystems. An interesting candidate would be Google's Android platform, which is the most installed mobile operating system in the world. In stark contrast to iOS, the ecosystem around Android is heavily fragmented with a great variety of manufacturers, smartphone hardware, and platform versions, which renders it especially prone to incompatible apps or devices. Assumingly, it is worthwhile to examine users' post-adoption behavior in connection to Android updates. Besides, future studies could reaffirm the discovery that updated operating systems can evoke multiple types of issues, each of which having a unique effect on sustained information systems usage. In a controlled laboratory experiment, participants could be confronted with errors in functionality, defective devices, or unsupported features consequential to a simulated platform update. Afterwards, they could be interviewed about their personal continuance intention with respect to each experienced problem category. This can help to reconfirm the validness of the observed results.

In conclusion, with the thesis at hand, a fundamental first step towards better understanding users' perception of operating system updates was made. The findings elucidated the hitherto poorly explored relationship between post-adoption behavior and platform evolution in mobile app ecosystems. Hence, this thesis laid foundations for further studies in a research area that will continue to grow in importance in the future.

Literature

Aggarwal, C. C., and Zhai, C. X. 2012. *Mining Text Data*, (Vol. 978-1–4614), New York: Springer Science & Business Media. (https://doi.org/10.1007/978-1-4614-3223-4).

Ahluwalia, R. 2002. "How Prevalent Is the Negativity Effect in Consumer Environments?," *Journal of Consumer Research* (29:2), pp. 270–279. (https://doi.org/10.1086/341576).

Amirpur, M., Fleischmann, M., Benlian, A., and Hess, T. 2015. "Keeping Software Users on Board - Increasing Continuance Intention through Incremental Feature Updates," in *European Conference on Information Systems*, pp. 1–16.

Anderson, E. W., and Sullivan, M. W. 1993. "The Antecedents and Consequences of Customer Satisfaction for Firms," *Marketing Science* (12:2), pp. 125–143. (https://doi.org/10.1287/mksc.12.2.125).

Appfigures Inc. 2019. "The Appfigures API," *API Documentation*. (https://docs.appfigures.com/, accessed January 15, 2020).

Apple Inc. 2013. "Apple's App Store Marks Historic 50 Billionth Download," *Apple Press Info*. (https://www.apple.com/newsroom/2013/05/16Apples-App-Store-Marks-Historic-50-Billionth-Download/, accessed December 23, 2019).

Apple Inc. 2018. "Ratings, Reviews, and Responses - App Store," *Apple Developer*. (https://developer.apple.com/app-store/ratings-and-reviews/, accessed January 12, 2020).

Apple Inc. 2019a. "Apple Celebrates the Best Apps and Games of 2019," *Apple Press Info*. (https://www.apple.com/newsroom/2019/12/apple-celebrates-the-best-apps-and-games-of-2019/, accessed March 12, 2020).

Apple Inc. 2019b. "About IOS 13 Updates," *Apple Press Info*. (https://support.apple.com/en-us/HT210393, accessed December 26, 2019).

Apple Inc. 2019c. "About an Issue That Impacts Third-Party Keyboard Apps in IOS 13 and IPadOS - Apple Support," *Apple Press Info*. (https://support.apple.com/en-us/HT210613?/en-us/advisory, accessed December 30, 2019).

Basole, R. C. 2009. "Visualization of Interfirm Relations in a Converging Mobile Ecosystem," *Journal of Information Technology* (24:2), pp. 144–159. (https://doi.org/10.1057/jit.2008.34).

Benlian, A., Koufaris, M., and Hess, T. 2011. "Service Quality in Software-as-a-Service: Developing the SaaS-Qual Measure and Examining Its Role in Usage Continuance," *Journal of Management Information Systems* (28:3), pp. 85–126. (https://doi.org/10.2753/MIS0742-1222280303).

Berente, N., Seidel, S., and Safadi, H. 2018. "Data-Driven Computationally Intensive Theory Development," *Information Systems Research* (30:1), pp. 50–64. (https://doi.org/10.1287/isre.2018.0774).

Bhattacherjee, A. 2001. “Understanding Information Systems Continuance: An Expectation-Confirmation Model,” *MIS Quarterly: Management Information Systems* (25:3), pp. 351–370. (https://doi.org/10.2307/3250921).

Bhattacherjee, A., and Premkumar, G. 2004. “Understanding Changes in Belief and Attitude toward Information Technology Usage: A Theoretical Model and Longitudinal Test,” *MIS Quarterly: Management Information Systems* (28:2), pp. 229–254. (https://doi.org/10.2307/25148634).

Bird, S., Klein, E., and Loper, E. 2009. *Natural Language Processing with Python*, Sebastopol: O’Reilly.

Blei, D. M., Ng, A. Y., and Jordan, M. I. 2003. “Latent Dirichlet Allocation,” *Journal of Machine Learning Research* (3:4–5), pp. 993–1022. (https://doi.org/10.1016/b978-0-12-411519-4.00006-9).

Bosch, J. 2009. “From Software Product Lines to Software Ecosystems,” in *Proceedings of the 13th International Software Product Line Conference*, pp. 111–119. (https://doi.org/10.1016/j.jss.2012.03.039).

Cavusoglu, Hasan, Cavusoglu, Huseyin, and Jun, Z. 2008. “Security Patch Management: Share the Burden or Share the Damage?,” *Management Science* (54:4), pp. 657–670. (https://doi.org/10.1287/mnsc.1070.0794).

Ceccagnoli, M., Forman, C., Huang, P., and Wu, D. J. 2012. “Cocreation of Valueina Platform Ecosystem : The Case of Enterprise Software,” *MIS Quarterly: Management Information Systems* (36:1), pp. 263–290. (https://doi.org/10.2307/41410417).

Chang, Y. P., and Zhu, D. H. 2012. “The Role of Perceived Social Capital and Flow Experience in Building Users’ Continuance Intention to Social Networking Sites in China,” *Computers in Human Behavior* (28:3), pp. 995–1001. (https://doi.org/10.1016/j.chb.2012.01.001).

Chen, N., Lin, J., Hoi, S. C. H., Xiao, X., and Zhang, B. 2014. “AR-Miner: Mining Informative Reviews for Developers from Mobile App Marketplace,” *International Conference on Software Engineering* (1), pp. 767–778. (https://doi.org/10.1145/2568225.2568263).

Chevalier, J. A., and Mayzlin, D. 2006. “The Effect of Word of Mouth on Sales: Online Book Reviews,” *Journal of Marketing Research* (43:3), pp. 345–354. (https://doi.org/10.1509/jmkr.43.3.345).

Claussen, J., Kretschmer, T., and Mayrhofer, P. 2013. “The Effects of Rewarding User Engagement: The Case of Facebook Apps,” *Information Systems Research* (24:1), pp. 186–200. (https://doi.org/10.1287/isre.1120.0467).

Davis, F. D., Bagozzi, R. P., and Warshaw, P. R. 1989. “User Acceptance of Computer Technology: A Comparison of Two Theoretical Models,” *Management Science* (35:8), pp. 982–1003. (https://doi.org/10.1287/mnsc.35.8.982).

Dittrich, K., and Duysters, G. 2007. "Networking as a Means to Strategy Change: The Case of Open Innovation in Mobile Telephony," *Journal of Product Innovation Management* (24:6), pp. 510–521. (https://doi.org/10.1111/j.1540-5885.2007.00268.x).

Eaton, B., Elaluf-Calderwood, S., Sørensen, C., and Yoo, Y. 2015. "Distributed Tuning of Boundary Resources: The Case of Apple's IOS Service System," *MIS Quarterly: Management Information Systems* (39:1), pp. 217–243.

Etzioni, O. 1996. "The World-Wide Web: Quagmire or Gold Mine?," *Communications of the ACM* (39:11), pp. 65–68.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. 1996. "From Data Mining to Knowledge Discovery in Databases," *AI Magazine* (17:3), pp. 37–53.

Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. 2014. "Do We Need Hundreds of Classifiers to Solve Real World Classification Problems?," *Journal of Machine Learning Research* (15), pp. 3133–3181. (https://doi.org/10.1117/1.JRS.11.015020).

Fleischmann, M., Amirpur, M., Grupp, T., Benlian, A., and Hess, T. 2016. "The Role of Software Updates in Information Systems Continuance - An Experimental Study from a User Perspective," *Decision Support Systems* (83), pp. 83–96. (https://doi.org/10.1016/j.dss.2015.12.010).

Fleischmann, M., Grupp, T., Amirpur, M., Hess, T., and Benlian, A. 2015. "Gains and Losses in Functionality-An Experimental Investigation of the Effect of Software Updates on Users' Continuance Intentions," *International Conference on Information Systems*, pp. 1–21.

Foerderer, J., and Heinzl, A. 2017. "Product Updates: Attracting New Consumers versus Alienating Existing Ones," *International Conference on Information Systems*, pp. 1–19. (https://doi.org/10.2139/ssrn.2872205).

Forbes. 2019. "Apple IOS 13 Is Full Of Bugs, Reports Warn," *Forbes Magazine*. (https://www.forbes.com/sites/gordonkelly/2019/09/19/apple-ios13-upgrade-problems-iphone-11-pro-max-xs-max-xr-update/#8e6952822bc4, accessed March 12, 2020).

Franzmann, D., Wiewiorra, L., and Holten, R. 2019. "Continuous Improvements: How Users Perceive Updates," *European Conference on Information Systems*, pp. 1–17.

Garg, R., and Telang, R. 2013. "Inferring App Demand from Publicly Available Data," *MIS Quarterly: Management Information Systems* (37:4), pp. 1253–1264. (https://doi.org/10.25300/MISQ/2013/37.4.12).

Genc-Nayebi, N., and Abran, A. 2017. "A Systematic Literature Review: Opinion Mining Studies from Mobile App Store User Reviews," *Journal of Systems and Software* (125), pp. 207–219. (https://doi.org/10.1016/j.jss.2016.11.027).

Ghazawneh, A., and Henfridsson, O. 2013. “Balancing Platform Control and External Contribution in Third-Party Development: The Boundary Resources Model,” *Information Systems Journal* (23:2), pp. 173–192. (https://doi.org/10.1111/j.1365-2575.2012.00406.x).

Grupp, T., and Schneider, D. 2017. “Seamless Updates - How Security and Feature Update Delivery Strategies Affect Continuance Intentions With Digital Applications,” *European Conference on Information Systems*, pp. 611–626.

Helal, S., Bose, R., and Li, W. 2012. *Mobile Platforms and Development Environments*, San Rafael: Morgan & Claypool Publishers. (https://doi.org/10.2200/s00404ed1v01y201202mpc009).

Herle, S. P., and Fan, G. 2010. “Apparatus and Method for Performing an Over-the-Air Software Update in a Dual Processor Mobile Station,” *U.S. Patent No. 7,810,088*, Washington DC: U.S.Patent and Trademark Office. (https://patents.google.com/patent/US8572597B2/en).

Hong, S. J., Thong, J. Y. L., and Tam, K. Y. 2006. “Understanding Continued Information Technology Usage Behavior: A Comparison of Three Models in the Context of Mobile Internet,” *Decision Support Systems* (42:3), pp. 1819–1834. (https://doi.org/10.1016/j.dss.2006.03.009).

Hong, W., Thong, J. Y. L., Chasalow, L., and Dhillon, G. 2011. “User Acceptance of Agile Information Systems: A Model and Empirical Test,” *Journal of Management Information Systems* (28:1), pp. 235–272. (https://doi.org/10.2753/MIS0742-1222280108).

Hutto, C. J., and Gilbert, E. 2014. “VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text,” in *Proceedings of the 8th International Conference on Weblogs and Social Media*, pp. 216–225.

Iansiti, M., and Levien, R. 2004. *The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability*, Boston: Harvard Business Press.

IDC Inc. 2020. “Smartphone OS Market Share,” *IDC Worldwide Quarterly Mobile Phone Tracker*. (http://www.idc.com/prodserv/smartphone-os-market-share.jsp, accessed March 14, 2020).

Jansen, S., Brinkkemper, S., and Finkelstein, A. 2009. “Business Network Management as a Survival Strategy: A Tale of Two Software Ecosystems,” *Proceedings of the First International Workshop on Software Ecosystems* (505:2), pp. 34–48.

Kahneman, D., and Tversky, A. 1979. “Prospect Theory: An Analysis of Decision under Risk,” *Econometrica* (47:2), pp. 263–91.

Krishnan, M. S., Mukhopadhyay, T., and Kriebel, C. H. 2004. “A Decision Model for Software Maintenance,” *Information Systems Research* (15:4), pp. 396–412. (https://doi.org/10.1287/isre.1040.0037).

Kruskal, W. H., and Wallis, W. A. 1952. “Use of Ranks in One-Criterion Variance Analysis,” *Journal of the American Statistical Association* (47:260), pp. 583–621. (https://doi.org/10.2307/2280779).

Kushwaha, A., Kumar Verma, S., and Sharma, C. 2012. “Analysis of the Concerns Associated with the Rapid Release Cycle,” *International Journal of Computer Applications* (52:12), pp. 20–25. (https://doi.org/10.5120/8254-1784).

Larsen, T. J., Sørebø, A. M., and Sørebø, Ø. 2009. “The Role of Task-Technology Fit as Users’ Motivation to Continue Information System Use,” *Computers in Human Behavior* (25:3), pp. 778–784. (https://doi.org/10.1016/j.chb.2009.02.006).

Liao, C., Lin, H. N., Luo, M. M., and Chea, S. 2017. “Factors Influencing Online Shoppers’ Repurchase Intentions: The Roles of Satisfaction and Regret,” *Information and Management* (54:5), pp. 651–668. (https://doi.org/10.1016/j.im.2016.12.005).

Limayem, M., Hirt, S. G., and Cheung, C. M. K. 2007. “How Habit Limits the Predictive Power of Intention: The Case of Information Systems Continuance,” *MIS Quarterly: Management Information Systems* (31:4), pp. 705–737. (https://doi.org/10.2307/25148817).

Manikas, K., and Hansen, K. M. 2013. “Software Ecosystems-A Systematic Literature Review,” *Journal of Systems and Software* (86:5), pp. 1294–1306. (https://doi.org/10.1016/j.jss.2012.12.026).

March, S. T., and Hevner, A. R. 2007. “Integrated Decision Support Systems: A Data Warehousing Perspective,” *Decision Support Systems* (43:3), pp. 1031–1043. (https://doi.org/10.1016/j.dss.2005.05.029).

Martin, W., Harman, M., Jia, Y., Sarro, F., and Zhang, Y. 2015. “The App Sampling Problem for App Store Mining,” *Conference on Mining Software Repositories* (2015-Augus), pp. 123–133. (https://doi.org/10.1109/MSR.2015.19).

McIlroy, S., Ali, N., Khalid, H., and Hassan, A. E. 2016. “Analyzing and Automatically Labelling the Types of User Issues That Are Raised in Mobile App Reviews,” *Empirical Software Engineering* (21:3), pp. 1067–1106. (https://doi.org/10.1007/s10664-015-9375-7).

McKinney, W. 2010. “Data Structures for Statistical Computing in Python,” *Proceedings of the 9th Python in Science Conference* (445), pp. 51–56.

Messerschmitt, D. G., and Szyperski, C. 2003. *Software Ecosystem: Understanding an Indispensable Technology and Industry*, Cambridge: MIT Press.

Moore, J. F. 1993. “Predators and Prey: A New Ecology of Competition.,” *Harvard Business Review* (71:3), pp. 75–86.

Moore, J. F. 1996. *The Death of Competition: Leadership and Strategy in the Age of Business Ecosystems*, New York: Harper Collins.

Ng, A. Y., and Jordan, M. I. 2002. “On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes,” in *Advances in Neural Information Processing Systems*, pp. 841–848.

Ng, B. Y., Kankanhalli, A., and Xu, Y. C. 2009. “Studying Users’ Computer Security Behavior: A Health Belief Perspective,” *Decision Support Systems* (46:4), pp. 815–825. (https://doi.org/10.1016/j.dss.2008.11.010).

Noei, E., Zhang, F., and Zou, Y. 2019. “Too Many User-Reviews - What Should App Developers Look at First?,” *IEEE Transactions on Software Engineering*, pp. 1–12. (https://doi.org/10.1109/tse.2019.2893171).

Oliver, R. L. 1977. “Effect of Expectation and Disconfirmation on Postexposure Product Evaluations: An Alternative Interpretation,” *Journal of Applied Psychology* (62:4), pp. 480–486. (https://doi.org/10.1037/0021-9010.62.4.480).

Oliver, R. L. 1980. “A Cognitive Model of the Antecedents and Consequences of Satisfaction Decisions,” *Journal of Marketing Research* (17:4), pp. 460–469. (https://doi.org/10.1177/002224378001700405).

Oliver, R. L. 1993. “Cognitive, Affective, and Attribute Bases of the Satisfaction Response,” *Journal of Consumer Research* (20:3), pp. 418–430. (https://doi.org/10.1086/209358).

Ortiz de Guinea, A., and Webster, J. 2013. “An Investigation of Information Systems Use Patterns: Technological Events as Triggers, the Effect of Time, and Consequences for Performance,” *MIS Quarterly: Management Information Systems* (37:4), pp. 1165–1188. (https://doi.org/10.25300/misq/2013/37.4.08).

Panichella, S., Di Sorbo, A., Guzman, E., Visaggio, C. A., Canfora, G., and Gall, H. C. 2015. “How Can i Improve My App? Classifying User Reviews for Software Maintenance and Evolution,” *Conference on Software Maintenance and Evolution* (1), pp. 281–290. (https://doi.org/10.1109/ICSM.2015.7332474).

Patterson, P. G. 1997. “Modeling the Determinants of Customer Satisfaction for Business-to-Business Professional Services,” *Journal of the Academy of Marketing Science* (25:1), pp. 4–17. (https://doi.org/10.1007/BF02894505).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, É. 2011. “Scikit-Learn: Machine Learning in Python,” *Journal of Machine Learning Research* (12), pp. 2825–2830.

Peppard, J., and Rylander, A. 2006. “From Value Chain to Value Network: Insights for Mobile Operators,” *European Management Journal* (24:2), pp. 128–141.

Qiu, Y., Gopal, A., and Hann, I. H. 2017. “Logic Pluralism in Mobile Platform Ecosystems: A Study of Indie App Developers on the IOS App Store,” *Information Systems Research* (28:2), pp. 225–249. (https://doi.org/10.1287/isre.2016.0664).

Recker, J. 2006. “Reasoning about Discontinuance of Information System Use,” *Journal of Information Technology* (17:1), pp. 21–31. (https://doi.org/10.1111/j.1467-8276.2007.00999.x).

Rehurek, R. 2019. “Gensim: Topic Modelling for Humans.” (https://radimrehurek.com/gensim/index.html, accessed February 2, 2020).

Rozin, P., and Royzman, E. B. 2001. “Negativity Bias, Negativity Dominance, and Contagion,” *Personality and Social Psychology Review* (5:4), pp. 296–320. (https://doi.org/10.1207/S15327957PSPR0504_2).

Ruhe, G., and Saliu, M. O. 2005. “The Art and Science of Software Release Planning,” *IEEE Software* (22:6), pp. 47–53. (https://doi.org/10.1109/MS.2005.164).

Scrapinghub Ltd. 2018. “Architecture Overview - Scrapy Documentation,” *Scrapy Developer Documentation*. (https://docs.scrapy.org/en/latest/topics/architecture.html, accessed January 14, 2020).

Scrapinghub Ltd. 2019. “Scrapy 1.8 Documentation,” *Scrapy Developer Documentation*. (https://docs.scrapy.org/en/latest/index.html, accessed January 14, 2020).

Sensor Tower Inc. 2019. “Q2 2019 - App Store Intelligence Data Digest,” *Data Digest Report*, pp. 1–54. (https://s3.amazonaws.com/sensortower-itunes/Quarterly+Reports/Sensor-Tower-Q2-2019-Data-Digest.pdf?src=landing, accessed January 13, 2020).

Sievert, C., and Shirley, K. 2014. “LDAvis: A Method for Visualizing and Interpreting Topics,” in *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, pp. 63–70. (https://doi.org/10.3115/v1/w14-3110).

Song, P., Xue, L., Rai, A., and Zhang, C. 2018. “The Ecosystem of Software Platform: A Study of Asymmetric Cross-Side Network Effects and Platform Governance,” *MIS Quarterly: Management Information Systems* (42:1), pp. 121–142. (https://doi.org/10.25300/MISQ/2018/13737).

Spearman, C. 1904. “The Proof and Measurement of Association between Two Things,” *American Journal of Psychology* (15:1), pp. 72–101. (https://doi.org/10.2307/1422689).

Stackpole, B., and Hanrion, P. 2007. *Software Deployment, Updating, and Patching*, Boca Raton: CRC Press. (https://doi.org/10.1201/9781420013290).

Tansley, A. G. 1935. “The Use and Abuse of Vegetational Concepts and Terms,” *Ecology* (16:3), pp. 284–307. (https://doi.org/10.2307/1930070).

Tiwana, A., Konsynski, B., and Bush, A. A. 2010. “Platform Evolution: Coevolution of Platform Architecture, Governance, and Environmental Dynamics,” *Information Systems Research* (21:4), pp. 675–687. (https://doi.org/10.1287/isre.1100.0323).

Venkatesh, V., Brown, S. A., and Bala, H. 2013. “Bridging the Qualitative-Quantitative Divide: Guidelines for Conducting Mixed Methods Research in Information Systems,” *MIS Quarterly: Management Information Systems* (37:1), pp. 21–54. (https://doi.org/10.25300/MISQ/2013/37.1.02).

Verhagen, T., Nauta, A., and Felberg, F. 2013. “Negative Online Word-of-Mouth: Behavioral Indicator or Emotional Release?,” *Computers in Human Behavior* (29:4), pp. 1430–1440. (https://doi.org/10.1016/j.chb.2013.01.043).

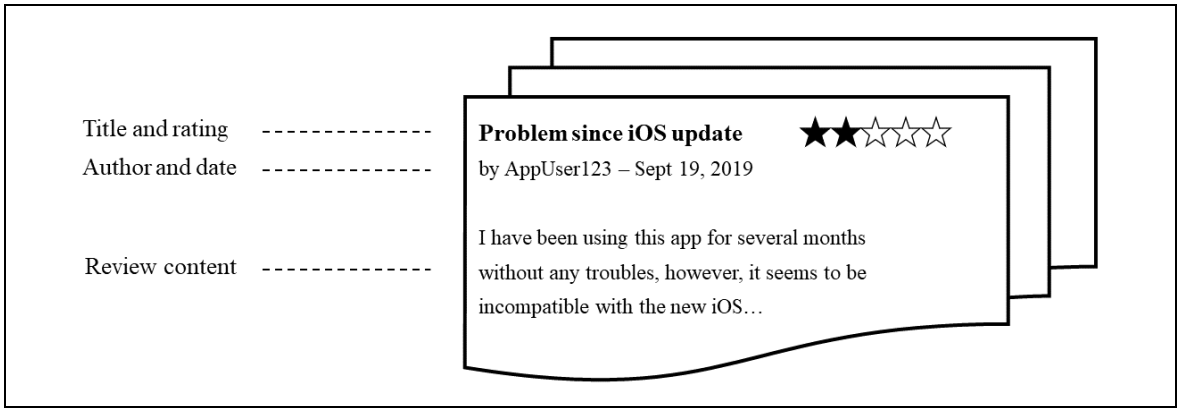
Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., and van Mulbregt, P. 2020. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, pp. 1–12. (https://doi.org/10.1038/s41592-019-0686-2).

Wang, H., Li, H., and Guo, Y. 2019. “Understanding the Evolution of Mobile App Ecosystems: A Longitudinal Measurement Study of Google Play,” *Proceedings of the World Wide Web Conference 2019*, pp. 1988–1999. (https://doi.org/10.1145/3308558.3313611).

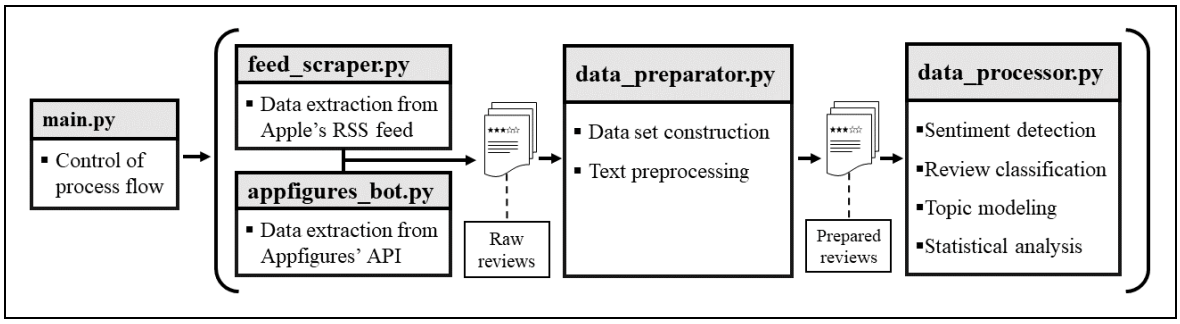
Williams, C. 2007. “Research Methods,” *Journal of Business & Economics Research* (5:3), p. 65.

Appendix

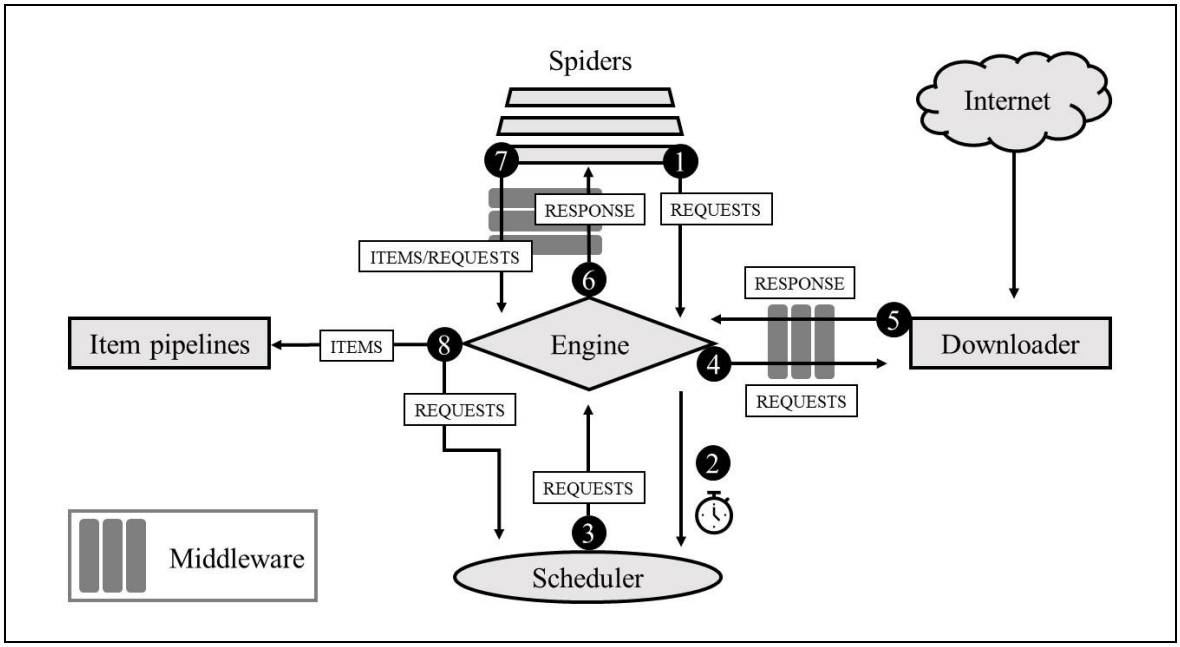
A Supplemental figures



Appendix A.1: Structure of App Store reviews



Appendix A.2: Overview of developed Python modules



Appendix A.3: Architecture of Scrapy (Scrapinghub Ltd. 2018)

B Supplemental tables

		Platform		
		Desktop	**Web**	**Mobile**
Category	**End user programming**	MS Excel, Mathematica, VHDL	Yahoo! Pipes, MS PopFly, Google's mashup editor	None so far
	Application	MS Office	SalesForce, eBay, Amazon, Ning	None so far
	Operating system	Windows, Linux, Apple OS X	Google AppEngine, Yahoo developer, Coghead, Bungee Labs	iOS, Android, Nokia S60, Palm

Appendix B.1: Software ecosystem taxonomy (Bosch 2009, p. 112)

Category	Rank	App	Publisher
Books	1	Audible audiobooks & originals	Audible, Inc.
Books	2	Amazon Kindle	AMZN Mobile LLC
Books	3	Wattpad - Books & Stories	Wattpad Corp
Books	4	KJV Bible -Audio Bible Offline	iDailybread Co., Limited
Books	5	Libby, by OverDrive	OverDrive, Inc.
Books	6	Yarn - Chat & Text Stories	Science Mobile, LLC
Books	7	OverDrive: eBooks & audiobooks	OverDrive, Inc.
Books	8	Goodreads: Book Reviews	Goodreads
Books	9	Color Therapy Coloring Number	Miinu Limited
Books	10	Hoopla Digital	Midwest Tape, LLC
Business	1	Indeed Job Search	Indeed Inc.
Business	2	ZOOM Cloud Meetings	Zoom
Business	3	ADP Mobile Solutions	ADP, Inc
Business	4	Duo Mobile	Duo Security
Business	5	Adobe Acrobat Reader for PDF	Adobe Inc.
Business	6	DoorDash - Driver	DoorDash, Inc.
Business	7	Amazon Flex	AMZN Mobile LLC
Business	8	Microsoft Teams	Microsoft Corporation
Business	9	ZipRecruiter Job Search	ZipRecruiter, Inc.
Business	10	Uber Driver	Uber Technologies, Inc.
Education	1	Photomath	Photomath, Inc.
Education	2	Google Classroom	Google LLC
Education	3	Remind: School Communication	remind101
Education	4	Quizlet	Quizlet Inc
Education	5	Mathway	Mathway, LLC
Education	6	Duolingo	Duolingo
Education	7	Kahoot! Play & Create Quizzes	Kahoot! AS
Education	8	Socratic by Google	Google LLC
Education	9	Elevate - Brain Training	Elevate, Inc.
Education	10	Slader Math Homework Answers	Slader, LLC
Entertainment	1	TikTok - Make Your Day	TikTok Inc.
Entertainment	2	Netflix	Netflix, Inc.
Entertainment	3	Hulu: Stream TV shows & movies	Hulu, LLC
Entertainment	4	Countdown App	Ryan Boyling
Entertainment	5	Amazon Prime Video	AMZN Mobile LLC
Entertainment	6	Tubi - Watch Movies & TV Shows	Tubi, Inc
Entertainment	7	Roku	ROKU INC
Entertainment	8	Ticketmaster - Buy, Sell Tickets	Ticketmaster
Entertainment	9	Slime Simulator Relax Games	ASMR - Slime Casual Games Studio
Entertainment	10	The CW	The CW Network
Finance	1	Cash App	Square, Inc.
Finance	2	Venmo: Send & Receive Money	Venmo
Finance	3	PayPal: Mobile Cash	PayPal, Inc.

Finance	4	Capital One Mobile	Capital One
Finance	5	Credit Karma	Credit Karma, Inc.
Finance	6	Zelle	Early Warning Services, LLC
Finance	7	Chase Mobile®	JPMorgan Chase & Co.
Finance	8	Bank of America Mobile Banking	Bank of America
Finance	9	Wells Fargo Mobile	Wells Fargo
Finance	10	Robinhood: Invest. Save. Earn.	Robinhood Markets, Inc.
Food & Drink	1	DoorDash - Food Delivery	DoorDash, Inc.
Food & Drink	2	Uber Eats: Food Delivery	Uber Technologies, Inc.
Food & Drink	3	Grubhub: Local Food Delivery	GrubHub.com
Food & Drink	4	Starbucks	Starbucks Coffee Company
Food & Drink	5	Popeyes®	Popeyes Louisiana Kitchen, Inc.
Food & Drink	6	Chick-fil-A	Chick-fil-A, Inc.
Food & Drink	7	McDonald's	McDonald's USA
Food & Drink	8	Postmates - Food Delivery	Postmates Inc.
Food & Drink	9	Domino's Pizza USA	Domino's Pizza LLC
Food & Drink	10	Dunkin'	Dunkin' Donuts
Games	1	Call of Duty®: Mobile	Activision Publishing, Inc.
Games	2	Photo Roulette	Photo Roulette AS
Games	3	Rescue Cut - Rope Puzzle	MarkApp Co. Ltd
Games	4	Fit the Ball 3D	Bigger Games
Games	5	Kolor it	ZPLAY
Games	6	Clash of Blocks!	Popcore GmbH
Games	7	Icing on the Cake	Lion Studios
Games	8	Brain Out	EYEWIND LIMITED
Games	9	Art Ball 3D	Alictus
Games	10	Mario Kart Tour	Nintendo Co., Ltd.
Health & Fitness	1	Calm	Calm.com
Health & Fitness	2	Reflectly	Reflectly
Health & Fitness	3	Flo Period & Ovulation Tracker	FLO HEALTH, INC.
Health & Fitness	4	MyFitnessPal	Under Armour, Inc.
Health & Fitness	5	Motivation Quotes -Daily Quote	Monkey Taps
Health & Fitness	6	BetterMen: Workout Trainer	Genesis Technology Partners
Health & Fitness	7	Sweatcoin	Sweatco Ltd
Health & Fitness	8	Headspace: Meditation & Sleep	Headspace Inc.
Health & Fitness	9	Fitbit	Fitbit, Inc.
Health & Fitness	10	Muscle Booster Workout Tracker	A.L. AMAZING APPS LIMITED
Lifestyle	1	Google Home	Google LLC
Lifestyle	2	Tinder	Tinder Inc.
Lifestyle	3	OnMyWay: Drive Safe, Get Paid	OnMyWay
Lifestyle	4	Zillow Real Estate & Rentals	Zillow.com
Lifestyle	5	Co-Star Personalized Astrology	Co-Star Astrology Society
Lifestyle	6	PINK Nation	Victoria's Secret PINK
Lifestyle	7	Bumble - Meet New People	Bumble Holding Limited
Lifestyle	8	Text Free: Texting + Calling	Pinger, Inc.
Lifestyle	9	T-Mobile Tuesdays	T-Mobile
Lifestyle	10	Hinge: Dating & Relationships	Hinge, Inc.
Photo & Video	1	YouTube: Watch, Listen, Stream	Google LLC
Photo & Video	2	Instagram	Instagram, Inc.
Photo & Video	3	Snapchat	Snap, Inc.
Photo & Video	4	Google Photos	Google LLC
Photo & Video	5	PicsArt Photo Editor + Collage	PicsArt, Inc.
Photo & Video	6	Triller: Social Video Platform	Triller LLC
Photo & Video	7	Polaroid Originals	Polaroid Originals
Photo & Video	8	Funimate Video Musical Editor	Avcr, Inc.
Photo & Video	9	Shutterfly: Cards & Gifts	Shutterfly
Photo & Video	10	Layout from Instagram	Instagram, Inc.
Productivity	1	Gmail - Email by Google	Google LLC

Productivity	2	Google Docs: Sync, Edit, Share	Google LLC
Productivity	3	Google Drive	Google LLC
Productivity	4	Google Slides	Google LLC
Productivity	5	Microsoft Outlook	Microsoft Corporation
Productivity	6	Google Sheets	Google LLC
Productivity	7	Dropbox	Dropbox
Productivity	8	Yahoo Mail - Organized Email	Yahoo
Productivity	9	CyberGhost VPN & WiFi Proxy	CyberGhost SRL
Productivity	10	Microsoft Word	Microsoft Corporation
Shopping	1	Amazon - Shopping made easy	AMZN Mobile LLC
Shopping	2	Walmart - Save Time and Money	Walmart
Shopping	3	Wish - Shopping Made Fun	ContextLogic Inc.
Shopping	4	eBay Shopping - Buy and Sell	eBay Inc.
Shopping	5	Target	Target
Shopping	6	Nike	Nike, Inc
Shopping	7	Poshmark	Poshmark, Inc.
Shopping	8	SHEIN-Fashion Shopping Online	Shein Group Ltd
Shopping	9	Arrive - Package Tracker	Shopify Inc.
Shopping	10	OfferUp - Buy. Sell. Simple.	OfferUp Inc.
Social Networking	1	Facebook	Facebook, Inc.
Social Networking	2	Messenger	Facebook, Inc.
Social Networking	3	WhatsApp Messenger	WhatsApp Inc.
Social Networking	4	Life360: Find Family & Friends	Life360
Social Networking	5	Pinterest	Pinterest
Social Networking	6	YOLO: Anonymous Q&A	Popshow, Inc.
Social Networking	7	IRL - Social Calendar	Live Awake Inc
Social Networking	8	Discord	Discord, Inc.
Social Networking	9	Google Duo	Google LLC
Social Networking	10	Monkey	Monkey, Inc.
Travel	1	Uber	Uber Technologies, Inc.
Travel	2	Lyft	Lyft, Inc.
Travel	3	Yelp Food & Services Around Me	Yelp
Travel	4	Airbnb	Airbnb, Inc.
Travel	5	Southwest Airlines	Southwest Airlines Co.
Travel	6	Google Earth	Google LLC
Travel	7	American Airlines	American Airlines
Travel	8	Expedia: Hotels, Flights & Car	Expedia, Inc.
Travel	9	GetUpside: Gas & Food Cashback	Upside Services Inc
Travel	10	United Airlines	United Airlines
Utilities	1	Google Chrome	Google LLC
Utilities	2	Google	Google LLC
Utilities	3	Fonts	Fonts LLC
Utilities	4	Bitmoji	Bitstrips
Utilities	5	My Verizon	Verizon Wireless
Utilities	6	myAT&T	AT&T Services, Inc.
Utilities	7	QR Reader for iPhone	TapMedia Ltd
Utilities	8	Ring - Always Home	Ring.com
Utilities	9	Verizon Call Filter	Verizon Wireless
Utilities	10	Microsoft Edge	Microsoft Corporation

Appendix B.2: Complete mobile app sample