Modeling Crowd Feedback in the App Market A Case Study on Food Delivery Apps

author(s)
affiliation(s)
address
email(s)

***Abstract*—Modern application (app) stores enable developers to classify their apps by choosing from a set of generic categories or genres such as *health*, *games*, and *music*. These categories are typically static—new categories do not necessarily emerge over time to reflect innovation in the mobile software landscape. With thousands of apps classified under each category, analyzing crowd feedback at such a vast scale can be a laborious and error-prone task. To overcome these limitations, in this paper, we propose an automated procedure for classifying and modeling crowd feedback within more focused categories of functionally-related mobile apps, or ecosystems. To realize this goal, we present a case study targeting the domain of food delivery apps. Specifically, our analysis in this paper is two-fold. First, using qualitative analysis methods, we synthesize important user concerns in the ecosystem of food delivery apps, and second, we propose and intrinsically evaluate an automated procedure for generating a succinct model of these concerns. Our generated model can help app developers to stay aware of the most pressing issues, or concerns, in their ecosystem, and thus, develop sustainable release engineering strategies that can respond to these issues in an effective and timely manner.**

***Index Terms*—software ecosystems, modeling, app store**

I. Introduction

Mobile application (app) stores have significantly expanded over the past decade to meet the growing demands of the mobile app market [1]. By the end of 2019, the Apple App Store (iOS app store) alone is expected to host close to two million active apps, growing by almost 2,000 apps per day [2]. To manage such a vast number of apps, app marketplaces classify apps into several broad categories (e.g., *Gaming*, *Health & Fitness*, and *Books*) and subcategories (e.g., *Sport*, *Board*, and *Card*) of generic application domains. This classification is intended to facilitate a more effective app discovery process and to help app developers discover potential rivals in their ecosystems [3].

Understanding the specific domain of competition is critical for app survival. Specifically, the clusters of functionally-related apps form distinct *micro-ecosystems* within the app store ecosystem. A software ecosystem can be defined 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 [4]. Systematically analyzing and synthesizing knowledge at such a micro level enables app developers to critically evaluate the current landscape of competition and to understand their end-users' expectations, preferences, and needs [5], [6], [7], [8], [9]. However, such knowledge is often tacit, embedded in the complex interplay between the human, system, and market components of the ecosystem. According to the organizational knowledge theory [10], to be effectively utilized, tacit domain knowledge must be translated into explicit form, or *externalized*. Once explicit knowledge is created, it can be preserved, used, and passed through to other individuals and organizations [11].

To address these challenges, in this paper, we propose an automated approach for modeling crowd concerns in micor-ecosystems (or *ecosystems* in short) of mobile apps. A user concern can be described as any direct or indirect, technical or nontechnical, behavior of the app that might negatively impact its users' experience or their overall well-being. The proposed approach is evaluated through a case study targeting the ecosystem of food courier, or delivery, apps. These apps, typically classified in popular app stores under the *Food & Drink* category, form a uniquely complex and dynamic ecosystem that consists of food consumers, drivers, and restaurants, operating in an extremely competitive environment and under strict business as well as technological constraints. The goal of our analysis is to demonstrate how such a complex ecosystem can be automatically analyzed and modeled. Specifically, our contributions in this paper can be described as follows:

- We qualitatively analyze a large dataset of user feedback collected from the Twitter feeds and app store reviews of four popular food delivery apps. Our objective is to identify the main pressing user concerns in the ecosystem of these apps.
- We propose, formally describe, and evaluate a fully automated procedure for modeling user concerns in the ecosystem along with their main attributes and triggers. The generated model is intended to provide app developers with a framework for assessing the fitness of their mobile apps and understanding the complex realities of their ecosystem.

The remainder of this paper is organized as follows. Section II motivates our research and presents our research questions. Section III describes our qualitative analysis. Section IV proposes an automated procedure for extracting and modeling crowd concerns in the ecosystem of food delivery apps. Section VI discusses our key findings and their impact. Section VII describes the study limitations. Finally, Section VIII concludes the paper and describes our future work.

II. Rationale and Research Questions

Apps are competing actors in an ecosystem of finite resources—only a handful of apps dominate downloads and revenue, leaving only a fraction of market value for other apps to compete over [1], [9], [13], [14]. In such a competitive market, end-user experience play a paramount role in the success of apps. Therefore, mobile app developers, trying to survive market selection, need to find effective mechanisms for monitoring crowd feedback and integrating that feedback into their release planning process [15], [16].

However, the majority of existing research on mining the crowd feedback in the mobile app market is focused on individual apps, with little attention paid to how such information can be utilized and integrated to facilitate software analysis at an ecosystem, or application domain, level [17], [18], [19]. Extracting concerns at such a level can be a more challenging problem than focusing on single apps, which typically receive only a limited number of reviews or tweets per day [20]. Furthermore, existing crowd feedback mining techniques are typically calibrated to extract technical user concerns, such as bug reports and feature requests, ignoring other, often non-technical, types of concerns that typically originate from the operational characteristics of the app. These observations emphasize the need for new methods that can integrate multiple heterogeneous sources of user feedback to reflect a more accurate picture of the ecosystem.

To bridge this gap in existing research, in this paper, we present a case study on modeling user feedback in ecosystems of functionally-related mobile apps. Our case study targets the ecosystem of food delivery, or courier, apps. The first major food courier service to emerge was Seamless, in 1999. A product of the internet boom, Seamless allowed users to order from participating restaurants using an online menu, a unique innovation that granted the service considerable popularity. Following Seamless, Grubhub was also met with success when it began offering web-based food delivery for the Chicago market in 2004. As smart phones became more popular, a number of new food couriers took advantage of the new demand for a more convenient mobile app-based delivery services. Of these competitors, UberEATS rose to the top, leveraging their experience with the mobile ride-share business model to adapt to food delivery. By the end of 2017, UberEATS became the most downloaded food-related app on the Apple App Store.

The domain of food delivery apps along with its users (e.g., restaurant patrons and drivers) and business (e.g., restaurants) components represent a uniquely complex and dynamic multi-agent software ecosystem. From a business point of view, these apps can be classified under the umbrella of *sharing economy*—a business model which leverages information technology to support peer-to-peer (P2P) based activities of acquiring, providing, or sharing access to goods and services in a community [21]. This unique business model imposes several challenges on the operational characteristics of food delivery apps. These challenges can be described as follows:

- **Fierce competition:** users often have multiple services to choose from within a given metropolitan area. Switching from one app to another is trivial, and users are highly impatient with late or incorrect orders. For instance, food delivery services have less than one hour for delivery. This forces developers to constantly innovate to provide faster delivery than their rivals.
- **Decentralized fulfillment:** the drivers are generally independent contractors who choose whom to work for and when to work. This creates challenges, not only for job assignment, but also for predicting when and where human resources will become available.
- **Multi-lateral communication:** in order to fulfill an order, the delivery app must communicate with users, drivers, and restaurants to ensure that the food order is ready when the driver arrives, and that the user knows when to expect delivery. Each channel of communication presents an opportunity for failure.

The complexity of these challenges makes the set of food delivery apps a particularly interesting subject to be targeted by our case study. Our main objective is to demonstrate the feasibility of automatically generating an abstract conceptual model of user concerns of such a dynamic ecosystem. Such model is intended to provide systematic technical and business insights for existing app developers as well as newcomers trying to break into the food delivery app market. To guide our analysis, we formulate the following research questions:

- ***RQ$_1$: What types of concerns are raised by users of food delivery apps?*** Mobile app users are highly vocal in sharing suggestions and criticism. Understanding this feedback is critical for evaluating and prioritizing potential changes to software. However, not all concerns, especially in business-oriented apps, are technical in nature. Therefore, developers must also be aware of business discussions, such as talk of competitors, poor service, or issues with other actors in their ecosystems.
- ***RQ$_2$: Can user concerns be automatically classified and modeled?*** Assuming that user feedback contains useful information, capturing, classifying, and modeling such information at an ecosystem level can be a challenging task. However, this type of analysis will provide valuable information for app developers, enabling them to discover the most important user concerns in their ecosystem as well as the main reasons that triggered these concerns.

III. User Concerns: Qualitative Analysis

To answer our first research question (**RQ$_1$**), in this section, we qualitatively analyze a large dataset of app store reviews and tweets, sampled from the crowd feedback of four popular food delivery apps. In what follows, we describe our data collection process as well as the main findings of our analysis.

A. Data Collection

In order to determine which apps should be considered for our ground truth dataset, we used the *top charts* feature of the Apple App Store and Google Play. Such charts keep the public aware of the top grossing and downloaded apps in the app store. UberEats is the most popular food delivery app on the App Store. Among the top ten apps in the *Food* category, there are three additional competing delivery apps: Doordash, GrubHub, and PostMates. If we broaden our focus to the top twenty-five apps, only one additional food delivery app is found, Eat24. Eat24 was recently acquired by GrubHub, and have redirected users to their parent app, allowing us to exclude it from the analysis. The Google Play Store shows the top 25 most popular apps in an arbitrary order rather than specific ranking. However, we find that UberEats and its three main competing apps are also present within the top 25. Therefore, the apps UberEats, Doordash, GrubHub, and PostMates will cover all the most popular food delivery services available on both platforms.

It is important to point out that, there are several, less popular, food delivery apps in the app market. These apps often operate in limited geographical areas (e.g., Waitr operating in southern US states) or have smaller user base. In our analysis, we are interested in apps operating in large geographical areas with the biggest market share, thus we narrowed down our ecosystem to its most *fit* elements from a user perspective. Popular apps receive significantly more crowd feedback on app stores and social media in comparison to smaller apps [20]. Furthermore, selecting mature apps gives smaller and newcomer apps a chance to learn from the mistakes of the big players in the market [22], [23].

After the list of apps is determined, the second step in our analysis is to identify and classify the main user concerns in the ecosystem. Prior research has revealed that software-relevant feedback can be found in tweets [24] and app store reviews [19], [25], [26], [27]. To extract reviews, we used the free third-party service *AppAnnie* [28]. This service allows reviews up to 90 days old to be retrieved from Google Play and the Apple App Store. To collect tweets, we limited our search to tweets directed to the Twitter account of the apps of interest. For example, to retrieve tweets associated with UberEats we searched for `to:ubereats`. Previous analysis has shown that this query form yields a large rate (roughly 50%) of meaningful technical feedback among the resulting tweets [29]. In our analysis, we scraped tweets directly from the Web page containing the results, going back for 90 days, covering September 4th to December 2nd of 2018.

In total, 1,833 tweets, 13,557 App Store reviews, and 29,674 Google Play reviews were extracted. Table I summarizes our dataset. Collecting data from multiple sources of feedback (multiple app stores and twitter) and over a long period of time is necessary to get the full picture of crowd feedback [30] and to minimize any sampling bias that might impact the validity of the analysis [31], [32].

B. Qualitative Analysis

To conduct our qualitative analysis, we sampled 900 posts (300 tweets, 300 iOS reviews, and 300 Android reviews) from the data collected for each app in our domain. To perform the sampling, we developed a Ruby program to first execute a `shuffle()` method on the lists of tweets and reviews to randomize the order. The first 300 posts from each source of user feedback were then selected.

TABLE I: Experimental Data

App	Tweets	Apple App Store	Google Play	Total
Doordash	344	6,685	5,273	12,302
GrubHub	414	1,058	2,863	4,335
Postmates	450	1,467	2,820	4,737
UberEats	625	4,347	18,718	23,690

To manually classify our data, we followed a systematic and iterative coding process. Specifically, three judges participated in the data classification process. The judges have multiple years of academic and industrial software engineering experience. For each post (tweet and review), each judge had to answer three main questions ***a)*** does the post raise any type of concern (informative vs. uninformative)?, ***b)*** what is the broad category of the concern?, and ***c)*** what is the specific concern raised in the post? The manual classification process was carried over 4 sessions, each session was 6 hours, divided into two periods of 3 hours each to avoid any fatigue issues and to ensure the integrity of the data [33]. The individually classified instances were then consolidated to generate the main categories of concerns as they appeared in the data. In what follows, we describe the results of our qualitative analysis in greater detail.

C. Results

A post during our manual classification task was considered *informative* if it raised any form of user concern. The rest of the reviews were considered miscellaneous. Post containing spam (e.g.,*"#UberEats Always late!! Check bit.ly/1xTaYs"*) or context-free praise or insults (e.g., *"I hate this app!"* and *"This app is great!"*) were also considered irrelevant. In general, the following categories and sub categories of concerns were identified in the set of informative posts:

- **Business concerns:** this category includes any concerns that are related directly to the business aspects of food delivery. In general, these concerns can be subdivided into two main subcategories:
  - **Human:** these concerns are related to interactions with employees of the apps. Users often complained about orders running late, cancellations, restaurant workers being rude, and drivers getting lost on the way to delivery. Human related reviews were on average the longest, often narrating multi-paragraph sequences of human (mainly driver) failures that led to undesirable outcomes.
  - **Market:** the apps in our dataset generally make money either through flat-rate delivery charges or surcharges added to the price of individual menu items. Users are highly sensitive to the differences between what they would pay at the restaurant versus at their doorstep. Posts complimenting low fees and markups were rare. Price complaints were

TABLE II: The number of posts (tweets and reviews) classified under each category of user feedback

	Tweets	iOS	And.	All
Human	443	649	276	1368
Market	392	563	340	1295
Business	704	931	522	2157
Bug	244	175	114	533
Feature	54	106	77	237
Technical	292	258	186	736

not the only form of market-related feedback. Other posts included generic discussions of market-related concerns such as business policy (such as refunds), discussion of competitors, promotions, and posts about participating restaurants and delivery zones. Requests for service in remote areas were fairly common.

- **Technical concerns:** this set of concerns includes any technical issues that are related to the user experience when using the app itself. As have been shown before [25], technical concerns often revolves around two subcategories:
  - **Bug reports:** posts classified under this category contain descriptions of software errors, or differences between a user's expectations and observed app behavior. Bug reports commonly consist of a simple narration of an app failure. In our dataset, we observe that the most common bugs relate to application of payment credits and app crashes.
  - **Feature requests:** these posts contain requests for specific functionality to be added to the app, or discussion of success/failure of distinct features. For example, some users of DoorDash complained about being forced to tip before the order was delivered. Users of Eat24 lament a recent update which removed the ability to reorder the last meal requested through the app. Under this category, we also include non-functional requirements (NFRs), or aspects of software which are related to overall utility of the app rather than its functional behavior (e.g., usability, reliability, security, and accessibility) [34], [35]. Ease-of-use was the most common NFR cited by users, followed by user experience (UX).

Table II shows the number of posts classified under each category of concerns in the sampled dataset. In general, our qualitative analysis revealed that, based on the total number of relevant posts, Android reviews were the least informative in comparison to other sources of feedback. One potential explanation for this phenomenon is that Google Play does not pose any restriction on the number of times an app can request users to leave a review for the app, while the Apple App Store limits app in this respect. As a result, many Android reviews were terse, with statements such as *"I'm only posting this because the app keeps nagging me"* being common. The results also show that the distribution of concerns over the apps was almost the same. As Fig. 1 shows, concern types spread almost equally among apps, highlighting the similarity between the apps in their core features and user base. It is important to point out that our identified categories were considered orthogonal: each post could be any combination of human, service, bug, and feature issues. Therefore, there was considerable overlap (e.g., Fig. 1) between categories.

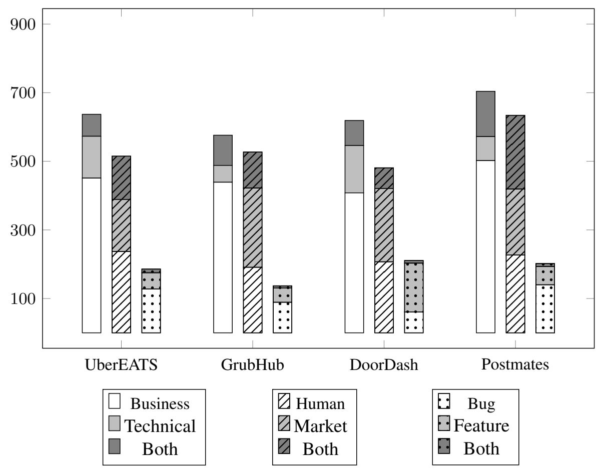

Fig. 1: The distribution of concern categories and subcategories for each app. Y-axes is the number of reviews.

In terms of specific concerns, nine different concerns were identified. Descriptions as well as examples of these concerns are shown in Table III.

IV. Modeling Crowd Feedback

In the first phase of our analysis, we qualitatively analyzed a large dataset of crowd feedback sampled from the set of app store reviews and tweets directed to the apps in our ecosystem. Our results showed that user concerns tend to be overlap and extend over a broad range of technical and business issues. Furthermore, these concerns tend to spread over multiple feedback channels and apps in the domain, which makes it practically infeasible to collect and synthesize such feedback manually. This emphasizes the need for automated tools that developers can use to make sense of such data.

Previous attempts to address these challenges through techniques such as topic modeling have been proven sub-optimal and often times impractical. Specifically, methods such as Latent Dirichlet Allocation (LDA) [58] requires calibrating several parameters and long *burin-in* periods in order to procedure cohesive topics. These topics typically need further processing (e.g., labeling, synthesizing) to be meaningfully interpreted [59], [60]. To overcome this challenge, in this section, we propose a practical and fully automated procedure for generating succinct representation of crowd feedback in mobile app ecosystems. In general, our model generation procedure can be divided into three main steps: ***a)*** classification: where informative user feedback is automatically captured, ***b)*** identifying entities: where important concepts in

TABLE III: Fine-grained classification of user concerns in the ecosystem of Food Delivery apps.

Concern	**Description**	**Example post**
Drivers	The single most common problem was with drivers. Specifically, drivers would be dispatched inefficiently, or combine orders, causing long wait times. Users were especially upset when drivers went the wrong direction.	*"In addition, the address that I gave to UberEats took the driver to a completely different parking lot"*, *"DoorDash The driver did not follow the order instructions, was belligerent, and shouted at me."*
Customer service	Users expressed dissatisfaction with service members friendliness of and how long it took to receive answers	*"Do not ever use this service! The contact number is nowhere to be found; I had to ask Google to find it."*
Refund	Users were often frustrated to discover that services would generally only offer refunds for the delivery charge, rather than for food, even if the food was rendered inedible due to long delivery time.	*"They were unable to get me a refund for food that arrived cold and rubbery when I live 3 minutes away from the restaurant."*
Service outage	Whenever the service is down, users immediately turn to social media to complain	*"The servers are down!"* and *"Great timing for an outage."*
Promo code	A common bug report was promotion codes not being applied to orders correctly	*"The promo code was rejected, inaccurately saying that I was not eligible."*
Communication	Bugs would sometimes stem from failed communication between the delivery service and the restaurant, especially regarding menu items and hours-of-operation.	*"Postmates so I ordered baby blues spent 52$ for my postmate to send me a picture of the place closed so I had to cancel my order and now I cant get food tonight."*
Security	Security errors were surprisingly common. Users would often find unexplained charges to their account.	*"Postmates my account was hacked. I reset my password and people all over the country are still ordering on my account."*
Routing	Occasionally the GPS systems in the drivers' apps would fail, causing the drivers to ask the user for help. Many users were upset when this happened.	*"Driver got lost had to ask me for BASIC directions, then drove in the complete opposite direction. The food came so late it was inedible."*
Order	Sometimes, services failed to route a driver to an order, and rather than alert the customer, they gradually pushed the delivery window back.	*"I had to contact grub hub, not the other way around, about a delivery that was an hour beyond the delivery window and the estimated time kept pushing further back."*

the feedback are identified, and ***c)*** estimating relationship: where the relationship between the model entities (concepts) are determined. In what follows, we describe these steps in greater detail.

A. Feedback Classification

The first step in our procedure is to classify our dataset into informative and uninformative user feedback. To classify our data, we experiment with three different classification algorithms: Support Vector Machines (SVM), Naive Bayes (NB), and Random Forests (RF). These algorithms have been extensively used to classify crowd feedback in the app market [19], [25], [36]. Their success can be attributed to their ability to deal effectively with short text (e.g., tweets, user reviews, YouTube comments, etc.) [37]. Our classification configurations can be described as follows:

- **Training settings:** To train our classifiers, we use 10-fold cross validation. This method creates 10 partitions of the dataset such that each partition has 90% of the instances as a training set and 10% as an evaluation set. The benefit of this technique is that it uses all the data for building the model, and the results often exhibit significantly less variance than those of simpler techniques such as the holdout method (e.g., 70% training set, 30% testing set).
- **Text pre-processing:** English stop-words were removed and stemming was applied to reduce words to their morphological roots. We use Weka's built-in stemmer and stop-word list to pre-process the posts in our dataset [38].
- **Sentiment Analysis:** we use sentiment analysis as a classification feature. The main assumption is that different categories of user concerns are expressed using different sentiments [39]. In our analysis, we used Sen-

TABLE IV: A comparison of the performance of the classifiers (SVM, NB, and RF) with lower-casing (LC), stemming (ST), stop-word (SW) removal, and sentiment analysis (SEN).

	NB			**SVM**			**RF**		
	P	R	F_2	P	R	F_2	P	R	F_2
Business									
LC	.85	.79	.82	.89	.85	.87	.85	.87	.86
LC + SEN	.85	.79	.82	.89	.85	.87	.86	.87	.86
LC + SW	.83	.84	.83	.89	.85	.87	.88	.86	.87
LC + SW + ST	.84	.83	.84	.89	.85	.87	.87	.88	.87
Technical									
LC	.38	.67	.49	.60	.55	.57	.93	.07	.13
LC + SEN	.38	.67	.49	.59	.55	.57	.95	.05	.10
LC + SW	.42	.69	.52	.58	.52	.55	.88	.22	.35
LC + SW + ST	.39	.71	.51	.61	.55	.58	.91	.16	.27

tiStrength [40]. SentiStrength assigns positive (p) and negative (n) sentiment scores for input text, based on the emotional polarity of indiviual words, using a scale of -5 to +5. To convert SentiStrength's numeric scores into these categories, we adapted the approach proposed by Jongeling et al. [41] and Thelwall et al. [42]. Specifically, a post is considered positive if $p + n > 0$, negative if $p + n < 0$, and neutral if $p + n = 0$.

The standard measures of Precision (P), Recall (R), and F-Measure (F_β) are used to evaluate the performance of our classification algorithms. Assuming t_p is the set of true positives, f_p is the set of false positives, and f_n is the set of false negatives. Precision is calculated as: $t_p/t_p + f_p$ and recall is calculated as: $t_p/t_p + f_n$. The F-measure is the weighted harmonic mean of P and R, calculated as $F_\beta = ((1 + \beta^2)PR)/(\beta^2 P + R)$. In our analysis, we use β = 2 to emphasize recall over precision.

All tweets and reviews in our original dataset were stored in ARFF format, a common text-based file format often used for representing machine learning datasets, and then fed to Weka, a data mining suite that implements a wide variety of machine learning and classification techniques[1]. Table IV shows the performance of NB, SVM, and RF in terms of P, R, and F_2. In general, stemming and stop-word removal made a small impact, and SVM provided the best average classification performance, achieving an F_2 of 0.65 in separating the different types of concerns, in comparison to an F_2 of 0.56 and 0.40 for NB and RF respectively. RF was evaluated with 100 iterations. Raising iterations above this number did not improve the performance.

In general, business-relevant posts were easier to classify than technically-relevant posts. This phenomenon is driven by the quantity of each class. Table II shows that technical posts were rare. The prior-probability of any given post being technical is less than 25%, negatively impacting the performance of all three classifiers. This problem was exacerbated for the individual technical categories, with features only occurring in 6.5% of posts. One of the reasons that technical posts were rare compared to other domains is that users had so many more business-related issues to discuss. Food courier services would often fail behind the scenes, causing drivers to be dispatched to incorrect locations, or customer support to fail to call. These failures often cause customers to discuss competition and pricing. As a result, business concerns *crowded out* technical concerns. In other domains, failures are more immediate and consumer visible, meaning that user concerns are more likely to take the form of bug reports.

Our results also show that the sentiment polarity of posts had almost no impact on the classification accuracy. In general, the results show that miscellaneous posts (posts not business or technically-relevant) were detected as having more positive sentiment than any other category. This result is expected; non-miscellaneous posts normally describe problems the user is having. Otherwise, the categories had substantially similar sentiment scores overall (Fig. 2).

B. Identifying model entities

In order to specify the main entities (nodes) of our model, we look for *important* words in the set of reviews and tweets classified as informative in the previous step. Our assumption is that such words capture the essence of user concerns in the ecosystem. In Object Oriented software design, when generating conceptual models from requirements text or any textual data, nouns are considered candidate classes (objects), verbs are considered as candidate operations (functions), while adjectives commonly represent attributes [43], [44]. Based on these assumptions, we only consider important nouns, verbs, and adjectives in our analysis.

To extract these parts of speech (POS), we utilize the Natural Language Toolkit (NTLK) [45] POS tagging library. We further apply *lemmatization* to reduce the morphological variants of words in our dataset down to their base forms. For example, *drink*, *drinks*, *drinking*, *drank*, and *drunk*, are all transformed to simply *drink*. By applying lemmatization, we avoid the problem of morphological variants being treated as entirely different words by our model. After lemmatization, we merge words together under each part of speech category. For example *drive* and *drives* are merged to simply *drive* when used as verbs. However, the word *drive* can also be a noun (e.g., *"that was a long drive"*). Therefore, we only merge words within the same part of speech to avoid losing this semantic distinction. Extracted parts of speech are then ranked based on their Hybrid $TF.IDF$ scores [46]. Formally, $TF.IDF$ can be computed as:

$$TF.IDF = TF(w_i) \times \lg \frac{|R|}{r_j : w_i \in r_j \wedge r_j \in R} \qquad (1)$$

where $TF(w_i)$ is the term frequency of the word w_i in the entire collection, $|R|$ is the total number of posts in the collection, and $r_j : w_i \in r_j \wedge r_j \in R$ is the number of posts in R that contain the word w_i. The purpose of $TF.IDF$ is to score the overall importance of a word to a particular document or dataset. If a word appears very rarely in general (low IDF), but very often in a small number of documents, then such a word will receive a very high $TF.IDF$ score.

[1]Data and script is provided and will be made public

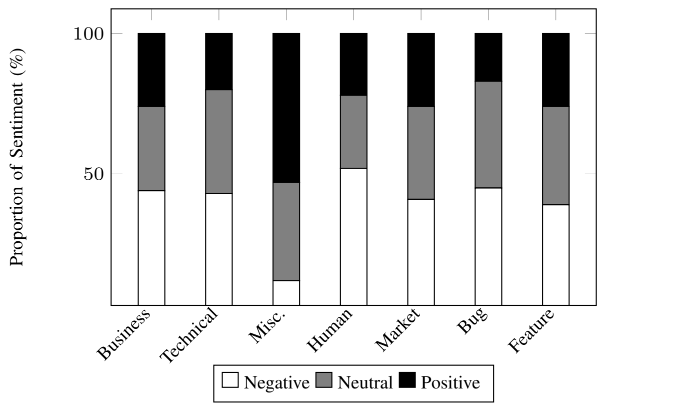

Fig. 2: The distribution of sentiment over the different types of posts.

Words with high scores on average are important to the dataset as a whole. After defining $TF.IDF$, we have the ability to extract important POS from the set of informative business and technical posts. The top 10 nouns in our dataset are {*food*, *app*, *service*, *delivery*, *time*, *consumer*, *driver*, *restaurant*, *money*}, the top 10 verbs are {*order*, *use*, *say*, *deliver*, *charge*, *wait*, *cancel*, *give*, *want*}, and the top 10 adjectives are {*good*, *great*, *terrible*, *horrible*, *wrong*, *last*, *bad*, *free*, *easy*}.

C. Identifying model relations

Our model generation procedure depends on the co-occurrence statistics of words in the data to capture their relations. For example, in our dataset, the words *customer* and *refund* appear in a very large number of user reviews and tweets. Therefore, the procedure assumes there is relation connecting these two entities. To count for such information, we use Pointwise Mutual Information (PMI).

PMI is an information-theoretic measure of information overlap, or statistical dependence, between two words [47]. PMI was introduced by Church and Hanks [47], and later used by Turney [48] to identify synonym pairs using Web search results. Formally, PMI between two words w_1 and w_2 can be measured as the probability of them occurring in the same text versus their probabilities of occurring separately. Assuming the collection contains N documents, PMI can be calculated as:

$$PMI = \log_2\left(\frac{\frac{C(w_1,w_2)}{N}}{\frac{C(w_1)}{N}\frac{C(w_2)}{N}}\right) = \log_2\left(\frac{P(w_1,w_2)}{P(w_1)P(w_2)}\right) \tag{2}$$

where $C(w_1, w_2)$ is the number of documents in the collection containing both w_1 and w_2, and $C(w_1)$, $C(w_2)$ are the numbers of documents containing w_1 and w_2 respectively. Mutual information compares the probability of observing w_1 and w_2 together against the probabilities of observing w_1 and w_2 independently. Formally, mutual information is a measure of how much the actual probability of a co-occurrence of an event $P(w_1, w_2)$ differ from the expectation based on the assumption of independence of $P(w_1)$ and $P(w_2)$ [49]. If the words w_1 and w_2 are frequently associated, the probability of observing w_1 and w_2 together will be much larger than the chance of observing them independently. This results in a PMI > 1. On the other hand, if there is absolutely no relation between w_1 and w_2, then the probability of observing w_1 and w_2 together will be much less than the probability of observing them independently (i.e., PMI < 1).

PMI is intuitive, scalable, and computationally efficient [50], [51]. These attributes have made PMI an appealing similarity method to be used to process massive corpora of textual data in tasks such as short-text retrieval [51], Semantic Web [48], [52], source code retrieval [53].

To generate the relations in our model, we computed the PMI between every pair of words to determine their relatedness. It is important to point out that, one potential pitfall of relying on PMI as a measure of relatedness, is that the highest PMIs will be found when one word only occurs a single time. This happens often with misspellings and irrelevant words. In order to prevent this phenomenon, we restrict our analysis to only words that occur at least ten times. Ten was chosen due to being the point at which sensitivity to additional increases became less noticeable (i.e., changing 10 to 11 would not substantially alter the results).

D. Model Representation

To generate our model, we extract the top 10 nouns ranked by $TF.IDF$ and then use PMI to extract the three most related verbs and adjectives with each noun. An example of a node, or an atomic entity in our model, is shown in Fig. 3. This node consists of three main parts:

- Concern: the middle part of the node represents the concern's name (*food*), which is basically one of the important nouns (based on $TF.IDF$) in our dataset.
- Properties: directly attached to the entity's name from the right is the top three adjectives associated with the entity (based on PMI). In our example, *food* could be *cold*, *hot*, or *ready*.
- Triggers: on the left side of the node, we attach the list the top three verbs frequently associated (based on PMI) with the noun (entity's name). Verbs often represent *triggers*, or leading causes of concerns. In our example, the verbs *arrive*, *deliver*, and *come* are commonly associated with the word *food*.

Formally, our model generation process can be described as follows, given a set of $Words$, containing all words in the dataset occurring at least ten times, we define $Adjs$, $Verbs$, and $Nouns$ as follows:

$$\begin{aligned} Adjs &= \{word \in Words \mid pos(word) = Adj\} \\ Verbs &= \{word \in Words \mid pos(word) = Verb\} \\ Nouns &= \{word \in Words \mid pos(word) = Noun\} \end{aligned} \tag{3}$$

We define three helper sets to help us express our graph mathematically. $SelNouns$ is the list of the top 10 selected nouns when ranked by Hybrid $TF.IDF$. $Verbs_w$ and $Adjs_w$ are the sets of three most closely related (by PMI) verbs and adjectives for a given words w. These sets are defined, using a function $top(n, pred)$ to retrieve the top n words after

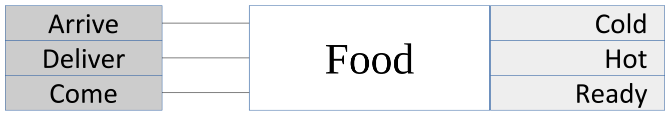

Fig. 3: The key elements of the entity-action-property relations represented by our model.

sorting each word by $pred(word)$. We use two functions to sort TF.IDF for nouns and PMI for verbs and adjectives. We express this using λ notation as follows:

$$\begin{aligned} SelNouns &= top(10, Nouns, \lambda x.TFIDF(x)) \\ Verbs_w &= top(3, Verbs, \lambda v.PMI(v, w)) \\ Adjs_w &= top(3, Adjs, \lambda a.PMI(a, w)) \end{aligned} \tag{4}$$

We define a graph, (V, E), expressed as a tuple of vertices and edges, as follows:

$$\begin{aligned} V = \bigcup \{w \in SelNouns \mid \{w\} \cup Verbs_w \cup Adjs_w\} \\ E = \{w \in SelNouns, v \in Verbs_w \mid (w, v)\} \cup \\ \{w \in SelNouns, a \in Adjs_w \mid (w, a)\} \end{aligned} \tag{5}$$

The set of vertices (V) is constructed by creating a smaller set containing each selected noun and its related adjectives and verbs, and then taking the union of these smaller sets to form the entire set of relevant entities, properties, and actions. The set of edges (E) is simply the union of associations of nouns to adjectives and nouns to verbs. Applying this process to our informative posts in the domain of food delivery apps results in the model in Fig. 4.

V. Model Evaluation and Interpretation

Due to the lack of *a priori* ground-truth, domain models evaluation can be a challenging task. In general, a domain model is an abstraction that describes a specific body of knowledge. Therefore, the quality of the model can be assessed based on its completeness, or it ability to encompass the main concepts present in the knowledge it models [54], [55]. These concepts are often determined manually by the domain experts. Based on these assumption, we examine the main concepts captured by our model. Specifically, In what follows, we measure the extent to which the noun-verb-adjective associations in our model reflect the main concerns identified in our qualitative analysis (Table III):

- **Drivers**: Drivers were a critical component of the ecosystem. All services struggled with driver *timing*, *directions*, and *friendliness*. Users complained about drivers *combining orders*. The model successfully identified <*driver, find*>, due to the presence of users discussing a driver's inability to find their house (therefore also <*driver, directions*>). The model's representation of the time estimates being inaccurate for drivers is captured in <*delivery, estimate*>. Lack of driver friendliness is captured in <*driver, awful*>.
- **Restaurants**: Restaurants were identified, both in terms of *selection* and *communication*. Users often asked services to add new restaurants, as well as discussed problems that occurred between the app, restaurant, and driver. The relations <*restaurant, show*> is expressed in the model partly due to users stating that the restaurant they want does not "show up" in the app. However, this phrase is more often associated with the *driver* not appearing at the restaurant. Communication is expressed with the <*restaurant, call*> association.
- **Customer Service**: Customer service was identified as an important concern when an *order* was not delivered on *time*, or *accurately*. Customers also complained about service in the context of not offering *refunds*, as well as giving general negative judgments of the overall customer service experience. In addition, customers sometimes struggled to *find* customer service numbers to begin with. The model identified both *customer* and *service* as important nouns and <*customer, refund*> was successfully identified along with <*customer, incorrect*> for accuracy. Both *customer* and *service* were associated with adjectives *poor* and *terrible* in the model. However, the ability to find numbers to call was not reflected.
- **Orders**: Orders were associated with *delays*. Users complained about receiving *cold food* as a result. Users were angered by food *waiting* at the restaurant to be picked up. The model identified <*order, leave*> in the context of dispatch and pickup times for orders. In addition, <*order, cold*> and <*order, hot*> were identified successfully.
- **Delivery**: Delivery was associated with a number of complaints about "slipping estimates" and incorrect "estimated times", explaining the relation <*delivery, estimate*> and <*delivery, estimated*>. The relation <*delivery, prepare*> occurred due to issues with orders being stuck in the preparation stage and never being dispatched for delivery. The relation <*delivery, choose*> primarily occurred in the context of users stating that they would *"choose a different delivery service"*.
- **Time**: Time was primarily found in complaints about delivery delays. The relations <*time, estimate*> and <*time, prepare*> appeared for the same reasons as the ones associated with *delivery*. A common occurrence was <*time, waste*> due to unexpected delays and order cancellations. The pair <*time, long*> occurred in similar contexts, as in *"it took longer than the estimated time"*.
- **Food**: Food was directly related to arrival as captured in the relations <*food, arrive*>, <*food, deliver*>, and <*food, come*>. Food was also associated with temperature, mainly due to the number of complaints about receiving cold food (e.g., <*food, cold*>). Complaints that orders had been sitting at the restaurant without being picked up by a driver were common, explaining the association relation <*food, ready*>.
- **App**: App appeared alongside comments about ease-of-use, resulting in <*app, easy*>. The relation <*app, ridiculous*> was a general complaint about poor policies or usability. The relation <*app, delete*> appeared when users discussed deleting an app after a poor experience.

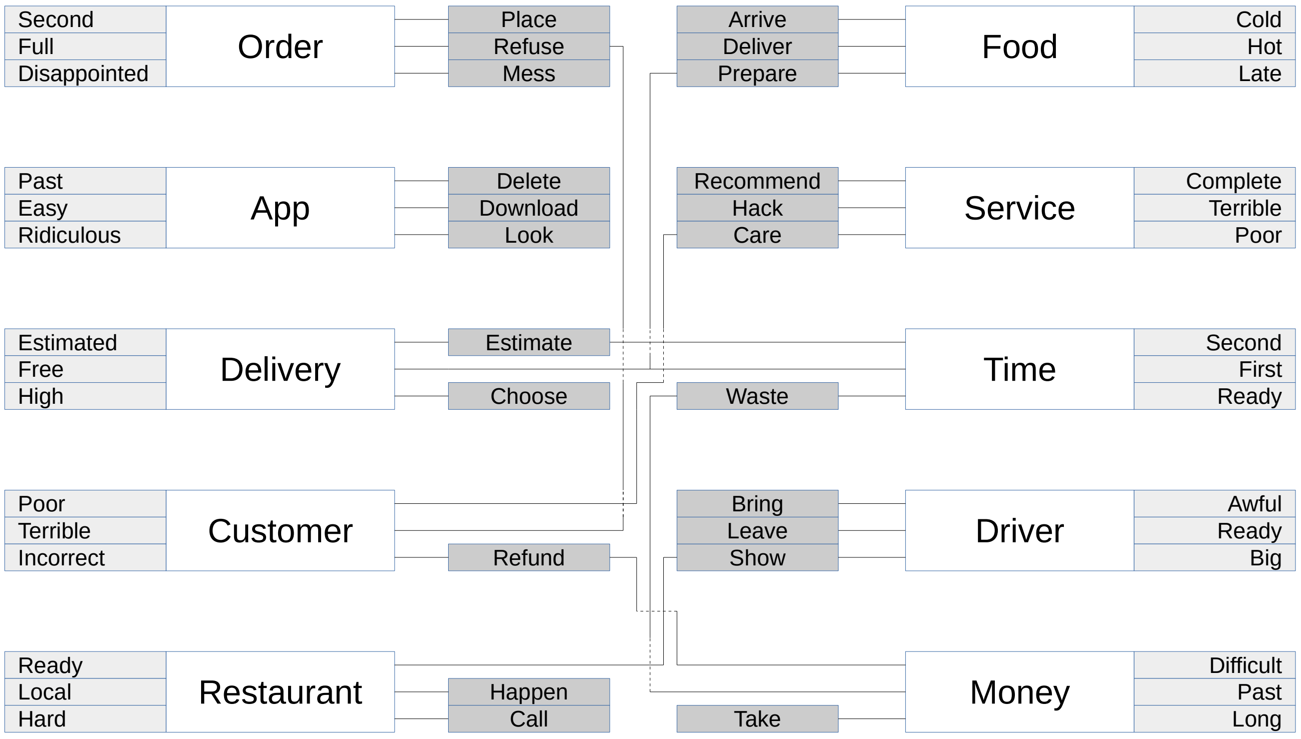

Fig. 4: A suggested model diagram depicting the relationships between important nouns (entities of the ecosystem), adjectives (attributes), and verbs (concern triggers), all extracted from business-relevant tweets.

A common association was <*app, look*>, appearing due to phrases such as *"look into this"* and *"looks like"*. The relation <*app, end*> appeared from users *"ended up"* eating cold or incorrect food, or not eating at all.

- **Money**: Money was captured in the relation <*money, waste*> mainly due to users ordering food that ended up being inedible and being unable to obtain a refund, also yielding the model relation <*money, refund*>. The verb *take* was associated with phrases like *"you take my money but did not deliver"* in the pair <*money, take*>.

In summary, in terms of completeness (the omission of domain concepts and the omission of relationships), our model was able to recover a large number of concepts in the data. Missed concerns were rare (e.g., inability to find customer service number). In terms of clarity, some of the captured relations (e.g., <*food, cold*> or <*money, waste*>) were more obvious than others (<*app, end*> or <*restaurant, show*>). However, the majority of captured relations were also identified during our qualitative analysis phase. Another observation is that technical concerns, despite not being accurately classified, have also found their way into the model. For instance, hacking was a popular technical concerns. The verb *hack* appeared in association with the nouns *customer* and *service*. In what follows, we discuss these results along with their potential impact.

VI. Discussion and Impact

The first phase of our analysis has revealed that user concerns in the app store are not necessarily always technical. In fact, our case study has exposed a new type of user concerns that extend beyond the technical issues of mobile apps, to cover other business and service oriented matters. These results emphasize the importance of studying user feedback in the app market at an ecosystem-level. Specifically, apps should be analyzed in bundles, or clusters, of functionally-related apps. In fact, such clusters can be automatically generated using app classification techniques [56]. Once these fine-grained categories are identified, automated data clustering, classification, and modeling techniques should be employed to consolidate and analyze user feedback to identify the main pressing user concerns in these clusters. Our analysis has also provided an additional evidence on the value of considering multiple sources of user feedback to get the full picture of user concerns [30].

In the second phase of our analysis, we proposed an automated procedure for generating a conceptual model of the main user concerns in our ecosystem. According to Eric Yu [57], *"conceptual modeling frameworks aim to offer succinct representations of certain aspects of complex realities through a small number of modeling constructs, with the intent that they can support some kinds of analysis"*. To that extent, our proposed work in this paper provided a first-of-its-kind procedure to automatically produce a conceptual model of crowd feedback in mobile app ecosystems. Our procedure adapted assertions from Object-Oriented programming and text processing to extract the main entities of our ecosystem. An underlying tenet is that the vocabulary of a domain provides an easily accessible supply of concepts. An information theoretic approach, which utilizes term co-occurrence statistics, was then used to establish a structuring mechanism for assembling and organizing extracted concepts. Our evaluation showed that relying on these techniques can generate a high quality model which captures most of the latent concepts in the domain knowledge. By changing the $TF.IDF$ and PMI thresholds and the number of nouns, verbs, and adjectives in Eq. 4, domain entities and relations can be included or excluded, thus, giving app developers the flexibility to generate domain models at different levels of granularity. The simplicity and configurability of our procedure gives it an advantage over other more computationally expensive methods, such as LDA [58], which often requires large amounts of data and the calibration of several hyperparameters in order to produce meaningful topics [59], [60].

Our generated model can provide valuable high-level and ecosystem-wide information to app developers, acting as a vehicle to facilitate the transition from domain knowledge to requirements specifications. For instance, startups, or newcomers, trying to break into the app market, can use our procedure to quickly generate a model for their new ecosystem of operation. Through the model entities and relations, they can get insights into the complex realities of their ecosystem. Such information can help them to redirect their effort toward innovations that can help to avoid these issues in their apps [15], [61], [62]. For example, in our specific ecosystem, developers can work on more accurate driver dispatching procedures to avoid delays, add new features for payments and refund to reduce amount of money and time wasted, add more security measures to prevent hacking, and implement smarter rating systems of drivers, customers, and restaurants, to control for the quality of service provided through the app. After release, developers can further use our model to automatically track users' reactions to their newly-released features.

VII. Limitations and Validity

Our analysis takes the form of a case study. Given that they target specific phenomena in their specific contexts, case studies often suffer from external validity threats [33]. For instance, our case study only included four apps. These apps might not represent the entire domain of food delivery. However, as mentioned earlier, our analysis was focused only on the most *fit* actors in the ecosystem. These popular apps often receive significantly more feedback than smaller apps [20]. Furthermore, to minimize any sampling bias, our data collection process included multiple sources of user feedback and has extended over long period of time to capture as much information about apps in our ecosystem as possible.

Internal validity threats may stem from the fact that, we only relied on the textual content of the reviews and their sentiment as classification features. In the literature, meta-data attributes, such as the star-rating of the review or the number of *retweets*, have also been considered as classification features [59], [18]. The decision to exclude such attributes was motivated by our goal of maintaining simplicity. Specifically, practitioners trying to use our procedure do not have to worry about collecting and normalizing such data, especially that the impact of such attributes on the quality of classification was found to be limited anyways [18], [59].

Threats might also stem from our model evaluation procedure. Specifically, our model was only evaluated intrinsically, based on how well the generated model correlated with the results of the qualitative analysis. While such evaluation can be sufficient for model generation and calibration tasks, it does not capture the practical significance of the model. Therefore, a main direction of future work will be dedicated to the extrinsic evaluation of our model. Extrinsic evaluation is concerned with criteria relating to the system's function, or role, in relation to its purpose (e.g., validation through experience). To conduct such analysis, our model will be provided to selected groups of app developers to be used as an integral part of their app development activities. Evaluation data will be collected through surveys that will measure the level of adaptation as well as the impact of such models on idea formulation and the success or failure of mobile app products.

VIII. Conclusions

In this paper, we proposed an automated approach for modeling crowed feedback in mobile app ecosystems. The proposed approach is evaluated through a case study targeting the ecosystem of the food delivery apps. Our results showed that users tend to express a variety of concerns in their feedback. Such concerns can extend over a broad range of issues, such as technical or business. The results also showed that, in our ecosystem of interest, business concerns were more prevalent than technical feedback. In the second phase of our analysis, we proposed an approach for automatically generating an abstract conceptual model of the main user concerns in our subject ecosystem. The results showed that a descriptive model could be generated by relying on the specificity, frequency, and co-occurrence statistics of nouns, verbs, and adjectives in textual user feedback. The results also showed that, despite being relatively rare and hard to classify, dominant technical concerns were reflected in the model.

In addition to extrinsically evaluating our model, our future work in this domain will include conducting more case studies, targeting apps operating in dynamic and multi-agent ecosystems, such as *Health & Fitness* or *Dating* apps. These models will be enriched with more information such as the priority of user concerns, or the magnitude/direction of the relation between two ecosystem entities. Such information will enable us to understand the mobile app market at a micro level and provide more succinct representations of its complex realities.

References

[1] T. Petsas, A. Papadogiannakis, M. Polychronakis, E. Markatos, and T. Karagiannis, "Rise of the planet of the apps: A systematic study of the mobile app ecosystem," in *Conference on Internet Measurement*, 2013, pp. 277–290.

[2] (2018) App stores - statistics and facts (https://www.statista.com/topics/1729/app-stores/). Statista.

[3] M. Yasini and G. Marchand, "Toward a use case based classification of mobile health applications," *Studies in health technology and informatics*, vol. 266, no. 210, pp. 175–179, 2015.

[4] S. Jansen, A. Finkelstein, and S. Brinkkemper, "A sense of community: A research agenda for software ecosystems," in *International Conference on Software Engineering - Companion Volume*, 2009, pp. 187–190.

[5] P. Coulton and W. Bamford, "Experimenting through mobile apps and app stores," *International Journal on Mobile Human Computer Interaction*, vol. 3, no. 4, pp. 55–70, 2011.

[6] A. Finkelstein, M. Harman, Y. Jia, W. Martin, F. Sarro, and Y. Zhang, "App store analysis: Mining app stores for relationships between customer, business and technical characteristics," University of College London, Tech. Rep. rN/14/10, 2014, Tech. Rep., 2014.

[7] M. Harman, Y. Jia, and Y. Zhang, "App store mining and analysis: MSR for app stores," in *Conference on Mining Software Repositories*, 108–111, p. 2012.

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

[9] Z. Svedic, "The effect of informational signals on mobile apps sales ranks across the globe," Ph.D. dissertation, School Business Faculty, Simon Fraser University, 2015.

[10] I. Nonaka, "A dynamic theory of organizational knowledge creation," *Organization Science*, vol. 5, no. 1, pp. 14–37, 1994.

[11] O. Glassey, "Method and instruments for modeling integrated knowledge," *Knowledge and Process Management*, vol. 15, no. 4, pp. 247–257, 2008.

[12] J. Gordon, "Creating knowledge maps by exploiting dependent relationships," *Knowledge Based Systems*, vol. 13, pp. 71–79, 2000.

[13] O. Carare, "The impact of bestseller rank on demand: Evidence from the app market," *International Economic Review*, vol. 53, no. 3, pp. 717–742, 2012.

[14] S. Lim and P. Bentley, "Investigating app store ranking algorithms using a simulation of mobile app ecosystems," in *Congress on Evolutionary Computation*, 2013, pp. 2672–2679.

[15] M. Nayebi, H. Farahi, and G. Ruhe, "Which version should be released to app store?" in *International Symposium on Empirical Software Engineering and Measurement*, 2017, pp. 324–333.

[16] M. Nayebi and G. Ruhe, "Optimized functionality for super mobile apps," in *International Requirements Engineering Conference*, 2017, pp. 388–393.

[17] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman, "A survey of app store analysis for software engineering," *Transactions on Software Engineering*, vol. 43, no. 9, pp. 817–847, 2017.

[18] W. Maalej, M. Nayebi, T. Johann, and G.Ruhe, "Toward data-driven requirements engineering," *IEEE Software*, vol. 33, no. 1, pp. 48–54, 2016.

[19] S. Panichella, A. Di Sorbo, E. Guzman, C. Visaggio, G. Canfora, and H. Gall, "How can I improve my app? Classifying user reviews for software maintenance and evolution," in *International Conference on Software Maintenance and Evolution*, 2015, pp. 767–778.

[20] S. Mcilroy, W. Shang, N. Ali, and A. Hassan, "User reviews of top mobile apps in apple and google app stores," *Communications of the ACM*, vol. 60, no. 11, pp. 62–67, 2017.

[21] G. Zervas, D. Proserpio, and J. Byers, "The rise of the sharing economy: Estimating the impact of airbnb on the hotel industry," *Journal of Marketing Research*, vol. 54, no. 5, pp. 687–705, 2017.

[22] S. McIlroy, W. Shang, N. Ali, and A. E. Hassan, "Is it worth responding to reviews? Studying the top free apps in google play," *IEEE Software*, vol. 34, no. 3, pp. 64–71, 2017.

[23] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," in *Requirements Engineering Conference*, 2013, pp. 125–134.

[24] E. Guzman, R. Alkadhi, and N. Seyff, "A needle in a haystack: What do Twitter users say about software?" in *International Requirements Engineering Conference*, 2016, pp. 96–105.

[25] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? On automatically classifying app reviews," in *Requirements Engineering Conference*, 2015, pp. 116–125.

[26] A. D. Sorbo, S. Panichella, C. Alexandru, J. Shimagaki, C. Visaggio, G. Canfora, and H. Gall, "What would users change in my app? Summarizing app reviews for recommending software changes," in *International Symposium on Foundations of Software Engineering*, 2016, pp. 499–510.

[27] M. Nayebi, H. Cho, H. Farrahi, and G. Ruhe, "App store mining is not enough," in *International Conference on Software Engineering Companion*, 2017, pp. 152–154.

[28] (2019) App annie (https://www.appannie.com/en/).

[29] G. Williams and A. Mahmoud, "Mining Twitter feeds for software user requirements," in *International Requirements Engineering Conference*, 2017, pp. 1–10.

[30] M. Nayebi, H. Cho, and G. Ruhe, "App store mining is not enough for app improvement," *Empirical Software Engineering*, vol. 23, no. 5, pp. 2764–2794, 2018.

[31] W. Martin, M. Harman, Y. Jia, F. Sarro, and Y. Zhang, "The app sampling problem for app store mining," in *Working Conference on Mining Software Repositories*, 2015, pp. 123–133.

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

[33] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslèn, *Experimentation in Software Engineering*. Springer, 2012.

[34] M. Glinz, "On non-functional requirements," in *Requirements Engineering*, 2007, pp. 21–26.

[35] J. Cleland-Huang, R. Settimi, O. BenKhadra, E. Berezhanskaya, and S. Christina, "Goal-centric traceability for managing non-functional requirements," in *International Conference on Software Engineering*, 2005, pp. 362–371.

[36] E. Guzman, M. El-Haliby, and B. Bruegge, "Ensemble methods for app review classification: An approach for software evolution," in *International Conference on Automated Software Engineering*, 2015, pp. 771–776.

[37] S. Wang and C. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Annual Meeting of the Association for Computational Linguistics*, 2012, pp. 90–94.

[38] J. Lovins, "Development of a stemming algorithm," *Mechanical Translation and Computational Linguistics*, vol. 11, pp. 22–31, 1968.

[39] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, M. Lanza, and R. Oliveto, "Sentiment analysis for software engineering: How far can we go?" in *International Conference on Software Engineering*, 2018.

[40] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas, "Sentiment strength detection in short informal text," *Journal of the Association for Information Science and Technology*, vol. 61, no. 12, pp. 2544–2558, 2010.

[41] R. Jongeling, P. Sarkar, S. Datta, and A. Serebrenik, "On negative results when using sentiment analysis tools for software engineering research," *Empirical Software Engineering*, vol. 22, no. 5, pp. 2543–2584, 2017.

[42] M. Thelwall, K. Buckley, and G. Paltoglou, "Sentiment strength detection for the social web," *Journal of the American Society for Information Science and Technology*, vol. 63, no. 1, pp. 163–173, 2012.

[43] R. Abbott, "Program design by informal English descriptions," *Communications of the ACM*, vol. 26, no. 11, pp. 882–894, 1983.

[44] M. Elbendak, P. Vickers, and N. Rossiter, "Parsed use case descriptions as a basis for object-oriented class model generation," *Journal of Systems and Software*, vol. 87, no. 7, pp. 1209–1223, 2011.

[45] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O'Reilly Media, 2009.

[46] D. Inouye and J. Kalita, "Comparing Twitter summarization algorithms for multiple post summaries," in *International Conference on Social Computing (SocialCom) and International Conference on Privacy, Security, Risk and Trust (PASSAT)*, 2011, pp. 298–306.

[47] K. Church and P. Hanks, "Word association norms, mutual information, and lexicography," *Computer Linguistics*, vol. 16, no. 1, pp. 22–29, 1990.

[48] P. Turney, "Mining the web for synonyms: PMI-IR versus LSA on TOEFL," in *European Conference on Machine Learning*, 2001, pp. 491–502.

[49] G. Bouma, "Normalized (pointwise) mutual information in collocation extraction," *German Society for Computation Linguistics and Language Technology*, pp. 31–40, 2009.

[50] D. Newman, Y. Noh, E. Talley, S. Karimi, and T. Baldwin, "Evaluating topic models for digital libraries," in *Annual Joint Conference on Digital Libraries*, 2010, pp. 215–224.

[51] R. Mihalcea, C. Corley, and C. Strapparava, "Corpus-based and knowledge-based measures of text semantic similarity," in *National Conference on Artificial Intelligence*, 2006, pp. 775–780.

[52] D. Sousa, L. Sarmento, and E. Rodrigues, "Characterization of the twitter replies network: Are user ties social or topical?" in *International Workshop on Search and Mining User-generated Contents*, 2010, pp. 63–70.

[53] S. Khatiwada, M. Tushev, and A. Mahmoud, "Just enough semantics: An information theoretic approach for IR-based software bug localization," *Information and Software Technology*, vol. 93, pp. 45–57, 2017.

[54] Rubén, "Domain analysis: An introduction."

[55] P. Mohagheghi and V. Dehlen, "Existing model metrics and relations to model quality," in *ICSE Workshop on Software Quality*, 2019, pp. 39–45.

[56] A. AlSubaihin, F. Sarro, S. Black, L. Capra, M. Harman, Y. Jia, and Y. Zhang, "Clustering mobile apps based on mined textual features," in *International Symposium on Empirical Software Engineering and Measurement*, 2016, pp. 38:1–38:10.

[57] E. S. Yu, "Conceptual modeling: Foundations and applications," A. T. Borgida, V. K. Chaudhri, P. Giorgini, and E. S. Yu, Eds. Springer-Verlag, 2009, ch. Social Modeling and I*, pp. 99–121.

[58] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[59] E. Guzman and W. Maalej, "How do users like this feature? A fine grained sentiment analysis of app reviews," in *Requirements Engineering*, 2014, pp. 153–162.

[60] N. Chen, J. Lin, S. Hoi, X. Xiao, and B. Zhang, "AR-Miner: Mining informative reviews for developers from mobile app marketplace," in *International Conference on Software Engineering*, 2014, pp. 767–778.

[61] M. Nayebi, B. Adams, and G. Ruhe, "Release practices for mobile apps – what do users and developers think?" in *International Conference on Software Analysis, Evolution, and Reengineering*, 2016, pp. 552–562.

[62] W. Martin, F. Sarro, and M. Harman, "Causal impact analysis for app releases in google play," in *International Symposium on Foundations of Software Engineering*, 2016, pp. 435–446.