Noname manuscript No.
(will be inserted by the editor)

Modeling User Concerns in Sharing Economy: The Case of Food Delivery Apps

Grant Williams · Miroslav Tushev · Fahimeh Ebrahimi · Anas Mahmoud

Received: date / Accepted: date

Abstract Sharing Economy apps, such as Uber, Airbnb, and TaskRabbit, have generated a substantial consumer interest over the past decade. The unique form of peer-to-peer business exchange these apps have enabled has been linked to significant levels of economic growth, helping people in resource-constrained communities to build social capital and move up the economic ladder. However, due to the multidimensional nature of their operational environments, and the lack of effective methods for capturing and describing their end-users' concerns, Sharing Economy apps often struggle to survive. To address these challenges, in this paper, we examine crowd feedback in ecosystems of Sharing Economy apps. Specifically, we present a case study targeting the ecosystem of food delivery apps. Using qualitative analysis methods, we synthesize important user concerns present in the Twitter feeds and app store reviews of these apps. We further propose and intrinsically evaluate an automated procedure for generating a succinct model of these concerns. Our work provides a first step toward building a full understanding of user needs in ecosystems of Sharing Economy apps. Our objective is to provide Sharing Economy app developers with systematic guidelines to help them maximize their market fitness and mitigate their end-users' concerns and optimize their experience.

Keywords Sharing Economy · Domain Modeling · Mobile Applications

1 Introduction

The recent decade has witnessed a major shift in the way people deal services and goods. This shift has been enabled by the emergence of a new form of busi-

Grant Williams
Microsoft
Redmond, Washington, 98052
E-mail: grwillia@microsoft.com

ness exchange, known as *Sharing Economy* (SE). Unlike conventional business models, SE is focused on providing access to— rather than ownership of— assets and resources via peer-2-peer (P2P) coordination [59]. This on-demand, convenient, and sustainable form of resource consumption has attracted consumers and investors around the globe. As of today, there are hundreds of SE platforms, enabling consumers to sell, rent, swap, lend, and borrow services and assets at unprecedented scales. According to PwC—the multinational professional services network—SE is projected to grow from 15 billion U.S. dollars in 2014 to close to 335 billion U.S. dollars by 2025 [74].

Earlier adaptations of *digital* SE can be traced back to the early days of the Internet. Ebay and Craigslist, both launched in 1995, are prominent examples of platforms that enabled a collaborative P2P circulation of services and assets. However, the real proliferation of SE can be attributed to mobile technology. Global networks of mobile devices has created an ideal environment for SE applications (apps) to thrive and reach the mainstream culture. Using mobile apps as a mediator, services such as Uber (ridesharing), Airbnb (lodging), and TaskRabbit (freelancing) paved the way for a new form of disruptive innovations that transformed the way we do business forever.

Despite their proven benefits, SE apps often struggle to survive. In addition to competing with each other (e.g., multiple apps providing the exact same service in the same geographical area), these apps have to compete with existing classical markets in ecosystems of finite resources (e.g., taxi, hotel, and retail industries) [28]. To survive this fierce competition, SE apps have to be in a constant state of innovation, driven by a deep knowledge of their end-users' expectations, preferences, and needs [27]. However, this knowledge is often tacit, embedded in the complex interplay between the user, system, and market components of the ecosystem of operation. In order to be effectively utilized, such knowledge must be translated into an explicit form, or *externalized* [70]. Once domain knowledge is externalized, it can be used to instantiate and sustain innovation [33].

To address these challenges, in this paper, we propose an automated approach for modeling crowd concerns in ecosystems of SE apps. We define a user concern as **any functional or non-functional behavior of the app that might negatively impact its users' experience or their overall well-being**. This abstract definition includes any technical (bugs or crashes) or nontechnical (service, economic, or social) issues that consumers of mobile apps may experience. The proposed approach is demonstrated 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 and technological constraints. The goal of our case study 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 prominent food delivery apps. Our objective is to understand and classify 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 of food delivery apps along with their main attributes and triggers. The generated model is intended to provide SE 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 2 provides a brief background of existing related research, motivates our work in this paper, and presents our research questions. Section 3 describes our qualitative analysis. Section 4 proposes an automated procedure for extracting and modeling crowd concerns in the ecosystem of food delivery apps. Section 5 discusses our key findings and their impact. Section 6 describes the main limitations of our study. Finally, Section 7 concludes the paper and describes our future work.

2 Background, Rationale, and Research Questions

In this section, we provide a brief summary of seminal related research, motivate our work, and present our research questions.

2.1 Background: Sharing Economy

The research on SE has become a prominent subject of research across multiple disciplines [27,45]. This can be explained based on the interdisciplinary nature of the problems often raised in this domain. In general, the research on SE can be categorized into five main categories:

- **Economic**: Recent research revealed that adapting solutions of SE can foster economic growth in big cities and local communities [20,100]. Specifically, SE can help to counter excessive spending and purchase habits [46] while generating new sources of revenue [62]. However, major concerns are frequently raised about the impact of this new business model on traditional long-established markets, affecting the revenue and business practices of these markets and threatening to put millions (e.g., taxi drivers and employees in the hotel industry) out of work by making their jobs obsolete [5,98].
- **Social**: Existing research often describe SE as a vehicle for building social capital and establishing social relationships within local communities [7, 87]. However, on the negative side, SE has paved the way for a new form of social challenges, including problems such as *digital discrimination*, which refers to scenarios where a business transaction is influenced by race, gender, age, or other aspects of appearance of service provider or receiver. [29].

For instance, a recent report by the National Bureau of Economic Research found that black riders using `Uber` waited 30% longer to be picked up [32]. Another study reported that non-black `Airbnb` hosts were able to charge 12% more than black hosts [29].

- **Environmental**: Several studies suggest that SE promotes environmental awareness by enabling more sustainable consumption practices in modern-day societies [3,13]. Other studies argue that this impact is not as substantial, suggesting that environmental factors are not as important for consumers as economic factors [2]. In fact, some other studies went even further to suggest that SE can lead to more environmental pressure and resource exploitation due to the more affordable alternatives it provides [87].
- **Legal**: This category of studies investigate existing regulations and suggest new regulatory infrastructures for protecting users of SE platforms from unwanted business practices. The main objective is to propose legislation to regulate the relationship between the app (e.g., Uber or Airbnb), service providers (e.g., drivers or apartment owners), and service receivers (e.g., riders or renters) [14,68], especially when the *terms-of-service* are somehow violated, such as in cases of drunk drivers, under-insured cars, unsafe apartments, and fraud [16].
- **Computing**: in computing, studies of SE often tackle the problem from an algorithmic and human–computer interaction (HCI) perspectives [27]. Algorithmic papers are mainly concerned with proposing new and more efficient algorithms for P2P matching, path planning in ride-sharing [21, 42], platform fairness [11,82,83], and pricing [11]. HCI related study, on the other hand, propose design solutions to optimize user experience [26], including protecting their privacy [35,95] and safety [6] and understanding their usage patterns and motivations to participate in SE [100].

2.2 Background: Mining Mobile App User Feedback

The research on mining mobile app user feedback has noticeably advanced in the past few years. The objective of this line of research is to help software developers infer their end-users' needs, detect bugs in their code, and plan for future releases of their apps. In general, two main channels of feedback are often considered: app store reviews and Twitter.

- **App store reviews:** A systematic survey of studies related to app store review analysis is provided in Martin et al. [61]. In general, this line of research proposes new tools (e.g., AR-Miner [19], MARA [47], MARC [50], and CLAP [89]), methods, and procedures for analyzing user reviews available on Google Play and the Apple App Store. The main objective is to capture any actionable maintenance requests in these reviews, such as bug reports and feature requests as well as non-functional requirements concerns, such as usability, reliability, security, and privacy [38,50].
  To automatically identify informative user reviews, reviews are typically classified using standard text classification techniques, including Naive

Bayes (NB), Support Vector Machines (SVM), Random Forests (RF), and Decision Trees (DT) [50,73] as well as clustering algorithms such as DBSCAN [89]. Simpler techniques, which rely on linguistic pattern and term matching have also been proposed in the literature [40,47,73]. In terms of modeling, techniques such as Latent Direchlet Allocation (LDA), are commonly used to infer meaningful high-level topics from reviews [19,40]. Text processing techniques, such as sentiment analysis, lemmatization, and part of speech tagging, are also commonly used to improve the accuracy of review classification and modeling techniques [17,58,64,91]. In addition, meta-data attributes of user reviews, such as their star rating and author information, are used to improve the predictive capabilities of review classifiers [54,58].

– **Twitter:** Twitter enables large populations of end-users of software to publicly share their experiences and concerns about their apps in the form of micro-blogs. Analysis of large datasets of tweets collected from the Twitter feeds of software systems revealed that around 50% of collected tweets contained actionable maintenance information [91]. Such information was found to be useful for different groups of technical and non-technical stakeholders [39], providing complementary information to support mobile app developers during release planning tasks. The results also showed that text classifiers, such as SVM and NB, summarization methods, such as Hybrid TF.IDF and SumBasic, and modeling methods, such as LDA, can be effectively used to categorize, summarize, and cluster software-related tweets into semantically related groups of technical feedback [91].

2.3 Research Gap and Motivation

Our review shows that systematically analyzing and synthesizing user feedback at a domain level can help app developers to critically evaluate the current landscape of competition and to understand their end-users' expectations, preferences, and needs [25,31,41,72,80]. Understanding the domain of competition is critical for the survival of SE apps. Specifically, the clusters of functionally-related SE 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 [49].

However, the majority of existing research on mining 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 [61,73]. Extracting concerns at a domain 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 [63]. Furthermore, existing crowd feedback mining techniques are calibrated to extract technical user concerns, such as bug reports and feature requests, often ignoring other non-technical types of concerns that originate

from the operational characteristics of the app [51,61]. 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 the gap in existing research** in this paper, we present a case study on modeling crowd feedback in ecosystems of SE apps. Our case study targets the ecosystem of food delivery apps. Emerging evidence has shown that, unlike other SE apps, the demand for food delivery services has significantly increased after the COVID-19 shelter-in-place order [18]. In fact, according to The New York Times, while use of Uber's ride-sharing service went down by 80% in April of 2020, UberEats has experienced 89% increase in demand [24]. This makes food delivery a particularly interesting SE domain to be targeted by our analysis.

2.4 Case Study Setup and Research Questions

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 ride-sharing to adapt to food delivery. By the end of 2017, UberEATS became the most downloaded food-related app on the Apple App Store.

The set of food delivery apps along with their consumer (e.g., restaurant patrons and drivers) and business (e.g., restaurants) components represent a uniquely complex and dynamic multi-agent ecosystem. This complexity imposes several challenges on the operation of these apps. These challenges, which can also be often found in other SE ecosystems, 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 main objective of our analysis is to demonstrate the feasibility of automatically generating an abstract conceptual model of user concerns in such a dynamic and complex ecosystem. Such model is intended to provide systematic technical and business insights for app developers as well as newcomers trying to break into the SE market. To guide our analysis, we formulate the two 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. Therefore, the first phase of our analysis is focused on systematically externalizing and classifying crowd feedback available in the Twitter feeds and app store reviews of food delivery apps.
- ***RQ$_2$: How can user concerns in the ecosystem of food delivery apps be automatically and effectively modeled?*** The second phase of our analysis is focused on automatically externalizing and modeling user concerns in the ecosystem of food delivery apps. Modeling such information can provide valuable information for SE app developers, enabling them to discover the most important user concerns in their ecosystem, along with their defining attributes and triggers. To answer this question, we propose an automated procedure for generating a new form of user feedback analysis models and we compare its performance to LDA, a commonly used technique for generating topics of app user concerns from online user feedback.

3 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.

3.1 Data Collection

In order to determine which apps to include in our case study, we used the *top charts* feature of the Apple App Store and Google Play. These charts keep the public aware of the top grossing and downloaded apps in the app store. As of September of 2018, 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[1]. The Google Play Store shows the top 25 most popular apps in an arbitrary order. 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 covers the most popular food delivery services available on both platforms.

It is important to point out that there are several other food delivery apps in the app market. These apps often operate in very limited geographical areas or have smaller user base. In our analysis, we are interested in apps with the biggest market share (as quantified by their app store download numbers), thus we narrowed down our ecosystem to its fittest elements from a user perspective. Popular apps receive significantly more crowd feedback on app stores and social media in comparison to smaller apps [63]. Furthermore, selecting mature apps gives smaller and newcomer apps a chance to learn from the mistakes of the big players in the market [71].

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 and app store reviews [58, 73, 77, 92]. To extract reviews, we used the free third-party service *AppAnnie*[2]. 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 our apps. For example, to retrieve tweets associated with UberEats, we searched for `to:ubereats`. Our previous analysis has revealed that this query form yields a large rate (roughly 50%) of meaningful technical feedback among the resulting tweets [92]. In our analysis, we collected tweets in the period from September 4$^{\text{th}}$ to December 4$^{\text{nd}}$ of 2018. In total, 1,833 tweets, 13,557 App Store reviews, and 29,674 Google Play reviews were extracted. Table 1 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 minimize any sampling bias that may impact the validity of the analysis [60].

3.2 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. Sampling 900 posts from the population of posts for each app ensures a confidence level of 99%. To perform the sampling, we developed a Ruby program to first execute a `shuffle()` method on the lists of tweets and

[1] `https://www.eat24.com/`

[2] `https://www.appannie.com/en/`

Table 1: Experimental Data, including the number of posts (tweets and reviews) collected from each channel of user feedback.

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

reviews to randomize the order, taking the time of the post into consideration to avoid selecting posts from the same time period (e.g., tweets from one week only). The first 300 posts from each source of user feedback were then selected.

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 an average of three years of industrial software engineering experience. For each post (tweet and review), each judge had to answer three main questions: ***a)*** does the post raise any concerns (informative vs. uninformative)?, ***b)*** what is the broad issue raised in the post?, and ***c)*** what is the specific concern raised in the post? The manual classification process was carried over four sessions, each session lasted around six hours, divided into two periods of three hours each to avoid any fatigue issues and to ensure the integrity of the data [94]. A final meeting was then held to generate the main categories of concerns as they appeared in the individually classified data. Conflicts were detected in less than 5% of the cases, mainly on the granularity level of the classification. For example, concerns about refunds and promo codes were considered two separate categories by one judge, while another judge classified them under the same concern category (money issues). Such conflicts were resolved after further discussion and eventually using majority voting. In what follows, we describe the results of our qualitative analysis in greater detail.

3.3 Results

A post during our manual classification task was considered *informative* if it raised any form of user concerns. The rest of the reviews were considered miscellaneous. Posts 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 general 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 (30 words), 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 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 too.

- **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 [58], technical concerns often revolve around two subcategories:
  - **Bug reports:** Posts classified under this category contain descriptions of software errors, or differences between the described and the observed behaviors of the app. Bug reports commonly consist of a simple narration of an app failure. In our dataset, we observed that the most common bugs were related to payments (174 out of 533) while crashes and service outages counted for 53 posts.
  - **Feature requests:** These posts contain requests for specific functionality to be added to the app, or discussions 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) [23, 34]. Ease-of-use was the most common NFR cited by users, followed by user experience (UX).

In terms of specific concerns, nine different concerns were identified: drivers, customer service, refund, service outage, promo code, communication, security, routing, and order. Thorough descriptions as well as examples of these concerns are shown in Table 2. In Table 3, we show 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

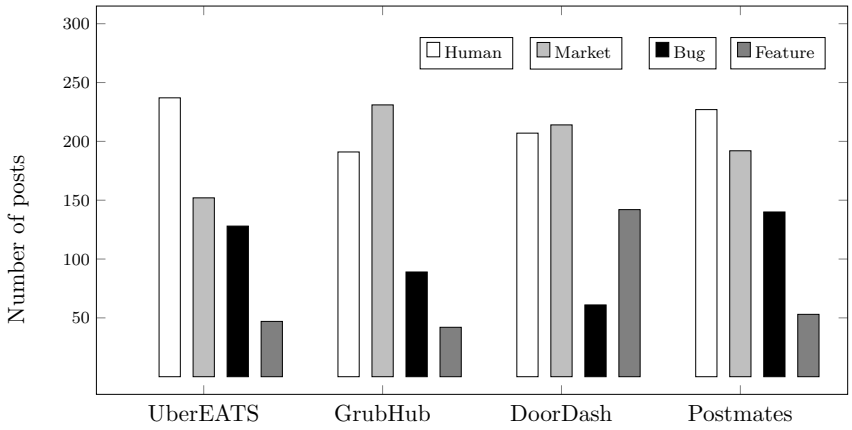


Fig. 1: The distribution of concern categories for each app. Y-axes is the number of posts (reviews and tweets).

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.

Finally, 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, market, bug, and feature issues. Therefore, there was considerable overlap between categories. This overlap is shown in Fig. 2.

4 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 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. To address these challenges, our second research question in this paper ($\mathbf{RQ_2}$) aims at proposing automated methods for generating representative models of the data. To answer this question, we first investigate the performance of LDA as one of the most commonly used topic modeling techniques in app user feedback analysis [19, 36, 40, 47]. We then propose a novel frequency-based approach for generating more expressive models of the data. The performance of both techniques is

Table 2: A 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 were dispatched inefficiently, or combined 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"* and *"@DoorDash The driver did not follow the order instructions, was belligerent, and shouted at me."*
Customer service	Users commonly expressed dissatisfaction with the friendliness of service members 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 generally only offered refunds for the delivery charge, excluding the order, 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 a service was down, users immediately turned 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 commonly originated 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. Several users reported unexplained charges to their accounts.	*"@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 failed, causing drivers to ask users 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 #grubhub, not the other way around, about a delivery that was an hour beyond the delivery window and the estimated time kept pushing further back."*

Table 3: The number of posts (tweets and reviews) classified under each category of user feedback. Length is the average number of words in the posts classified under each category.

	Tweets	**iOS**	**Android**	**Total**	**Length**
Human	443	649	276	1368	30
Market	392	563	340	1295	28
Business	704	931	522	2157	27
Bug	244	175	114	533	27
Feature	54	106	77	237	24
Technical	292	258	186	736	25

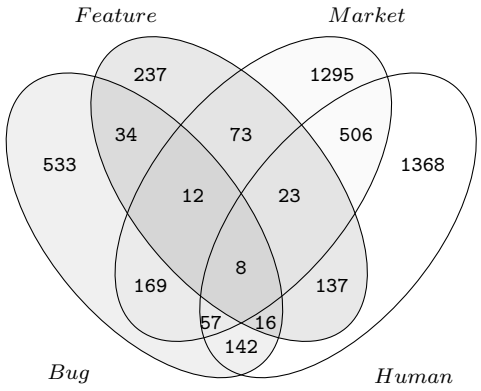


Fig. 2: A Venn diagram of the distribution of classification labels and their overlap in the dataset. For example, the diagram shows that there is 1368 unique Human-related concerns. Among those, 506 were also classified as Market concerns, 23 as Human, Market, and Feature, 137 as Human and Feature, 16 as Human, Bug, and Feature, 57 as Human, Market, and Bug, 142 as Human and Bug, and 8 as Human, Market, Feature, and Bug.

evaluated based on their ability to capture the main concerns of food delivery app users as well as their main attributes and triggers (Table 2).

4.1 Baseline: Modeling User Concerns with LDA

Introduced by Blei et al. [12], LDA is an unsupervised probabilistic approach for estimating a topic distribution over a text corpus. A topic consists of a group of words that collectively represents a *potential* thematic concept [12, 43]. Formally, LDA assumes that words within documents are the observed data. The known parameters of the model include the number of topics k, and the Dirichlet priors on the topic-word and document-topic distributions β and α. Each topic t_i in the latent topic space ($t_i \in T$) is modeled as a multi-dimensional probability distribution, sampled from a Dirichlet distribu-

tion β, over the set of unique words ($w_i \in W$) in the corpus D, such that, $\phi_{w|t} \sim Dirichlet(\beta)$. Similarly, each document from the collection ($d_i \in D$) is modeled as a probability distribution, sampled from a Dirichlet distribution α over the set of topics, such that, $\theta_{t|d} \sim Dirichlet(\alpha)$. $\theta_{t|d}$ and $\phi_{w|t}$ are inferred using approximate inference techniques such as Gibbs Sampling [37]. Gibbs sampling creates an initial, naturally weak, full assignment of words and documents to topics. The sampling process then iterates through each word in each document until word and topic assignments converge to an acceptable (stable) estimation [12].

4.1.1 Topic Extraction

We use Gensim[3] to extract topics from our dataset of user posts (reviews and tweets) [75]. Gensim is a Python-based open-source toolkit for vector space modeling and topic modeling. We apply lemmatization and stop-word removal on the posts to enhance the quality of generated topics. For lemmatization we use the spaCy library for Python[4] and to remove stop-words we use Gensim's built-in stop-word removal function. LDA's hyper-parameters α and β are optimized by Gensim, where α is automatically learned from the corpus and β is set to be 1/(number of topics). To determine the number of topics, we rely on Gensim's coherence score. Topic coherence provides a convenient measure to judge how good a given topic model is. Our analysis shows that at around 8-10 topics, our data will generate the most cohesive topics (Fig. 3-a).

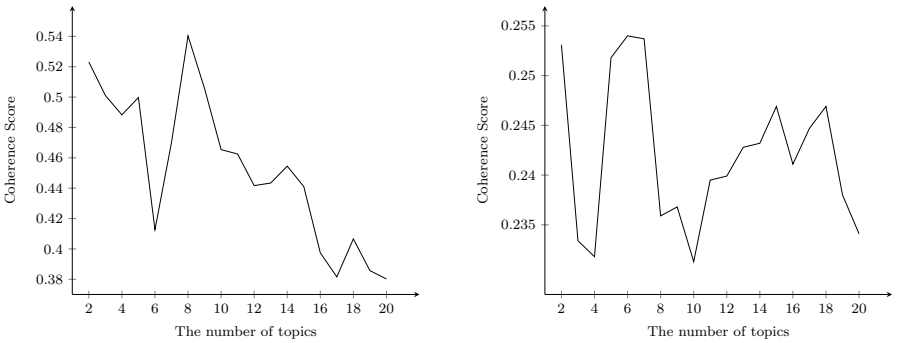
4.1.2 Results

The list of generated topics are shown in Table 4. In general, the topics are of poor quality, in other words, they do not seem to capture any of the major concerns identified either by our qualitative analysis. For example, while the second topic in Table 4 includes words such as *delivery*, *food*, and *fee*, it fails to represent a coherent concern due to the mixture of words from more than one concern category. Other topics in Table 4 also contain almost no words collectively representative of any of the concern categories identified during our qualitative analysis phase.

These poor results can be explained based on the limited length of user reviews and tweets. Recent research has shown that LDA does not perform well when the input documents are short in length [9,44,96]. Specifically, LDA is a data-intensive technique that requires large quantities of text to generate meaningful topic distributions. However, due to the sparsity attribute of short-text, applying standard LDA to short-text data (e.g., user reviews or tweets) often produces incoherent topics [44,99]. To overcome this problem, researchers use supplemental strategies to effectively train LDA in short-text environments. Such strategies, often known as *pooling*, are based on merging

[3] `https://radimrehurek.com/gensim/`

[4] `https://spacy.io`



a) LDA b) Assisted LDA

Fig. 3: The impact of the number of topics on the coherence score.

(aggregating) related texts together and presenting them as single *pseudo-documents* to LDA, thus, increasing the amount of text per document to work with. In our analysis, we aggregate posts from each source (App Store reviews, Google Play reviews, and Twitter) for each app in a single document, thus producing 3 x 4 documents. We then generate topics for our aggregated data. Using this data, the coherence score hits a local maxima at 6 topics (Fig. 3-b). The generated topics are shown in Table 5.

In general, aggregating user posts resulted in producing very similar topics. Generated topics are more redundant, providing only incomplete representations of the user concern in our data. The poor generalization ability of LDA can be attributed to two main reasons. First, due to the overlapping nature of the different concern categories, the classes are not separable by LDA. As a result, we see a mixture of words from different concern categories in the same topic. Second, LDA is a data-intensive technique that requires large quantities of text to generate meaningful topic distributions [12]. However, our dataset is relatively small, consisting of only 3,600 user posts, and even much less documents when these posts are aggregated.

In summary, our attempt to automatically generate our list of concerns using LDA was relatively unsuccessful. In order to generate meaningful topics, LDA requires a balance between the number and length of text artifacts being modeled [81]. While we had a relativity large number of artifacts, their length was limited. Our attempt to generate larger artifacts using Assisted LDA resulted in only few lengthy artifacts (12). This has negatively impacted LDA's ability to converge, or generate meaningful latent topic structures. Our expectation is that, applying more fine-grained text aggregation strategies that can produce sufficiently long, but not too long, documents (e.g., aggregating tweets based on hashtags) would help to improve the quality of generated topics [44, 66].

Table 4: Topics generated by LDA for our dataset of use feedback.

Topic 1		**Topic 2**		**Topic 3**		**Topic 4**	
Word	**Prob.**	**Word**	**Prob.**	**Word**	**Prob.**	**Word**	**Prob.**
uber	0.078	delivery	0.092	option	0.076	well	0.077
nice	0.036	food	0.073	horrible	0.055	terrible	0.066
number	0.021	even	0.034	location	0.039	fast	0.028
see	0.032	fee	0.031	home	0.028	actually	0.027
payment	0.025	love	0.026	dollar	0.028	be	0.018
super	0.019	pay	0.024	live	0.026	speak	0.018
unable	0.018	go	0.020	night	0.022	life	0.016
check	0.017	want	0.020	download	0.018	report	0.014
user	0.017	come	0.019	awful	0.014	name	0.014
many	0.016	cold	0.015	part	0.014	buy	0.013
Topic 5		**Topic 6**		**Topic 7**		**Topic 8**	
Word	**Prob.**	**Word**	**Prob.**	**Word**	**Prob.**	**Word**	**Prob.**
ever	0.081	order	0.090	good	0.186	app	0.127
company	0.051	service	0.045	card	0.052	great	0.033
mobile	0.022	get	0.044	awesome	0.038	restaurant	0.032
steal	0.021	time	0.037	use	0.038	try	0.028
apply	0.021	customer	0.030	everything	0.025	uber-eats	0.025
wish	0.017	food	0.022	think	0.024	work	0.024
create	0.016	say	0.021	sign	0.022	place	0.023
quick	0.015	driver	0.020	next	0.017	use	0.018
scam	0.014	never	0.019	amazing	0.017	go	0.018
basically	0.014	call	0.019	delay	0.015	way	0.017

4.2 Proposed Modeling Approach

The first part of our modeling analysis showed that LDA comes with several inherent limitations related to its computational complexity and the nature of our data. These limitations prevent LDA from producing meaningful representations of crowd feedback. To overcome these limitations, in this section, we propose a fully automated procedure for generating succinct representations of crowd feedback in the ecosystem of food delivery apps. In general, our automated model generation procedure can be divided into four main steps:

1. Informative feedback is captured.
2. Important concepts (domain entities) in the feedback are identified.
3. Relationships between the domain entities are determined.
4. Entities and relations are consolidated to automatically generate the model.

In what follows, we describe these steps in greater detail.

4.2.1 Identifying Informative Feedback

The first step in our procedure is to separate informative user feedback from uninformative feedback. A large body of research exists on classifying mobile

Table 5: Topics generated by Assisted LDA for our dataset of user feedback.

Topic 1		**Topic 2**		**Topic 3**	
Word	**Prob.**	**Word**	**Prob.**	**Word**	**Prob.**
order	0.002	order	0.042	app	0.037
app	0.001	food	0.022	order	0.030
food	0.001	app	0.020	food	0.023
get	0.001	get	0.019	great	0.020
service	0.001	service	0.019	time	0.018
delivery	0.001	time	0.017	love	0.017
driver	0.001	delivery	0.016	delivery	0.017
time	0.001	customer	0.012	get	0.015
restaurant	0.001	driver	0.012	service	0.015
customer	0.001	never	0.010	good	0.014
Topic 4		**Topic 5**		**Topic 6**	
Word	**Prob.**	**Word**	**Prob.**	**Word**	**Prob.**
order	0.002	order	0.023	app	0.037
app	0.001	get	0.021	good	0.030
food	0.001	food	0.015	order	0.023
get	0.001	app	0.012	food	0.020
delivery	0.001	service	0.011	delivery	0.018
service	0.001	driver	0.009	service	0.017
customer	0.001	customer	0.009	nice	0.017
time	0.001	guy	0.009	time	0.015
driver	0.001	doordash_help	0.008	get	0.015
never	0.001	delivery	0.008	bad	0.014

app user feedback into different categories of software maintenance tasks, such as feature requests and bug reports [58, 73, 92]. Our classification configurations can be described as follows:

- **Classification algorithms:** To represent 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 [58, 73]. Their success can be attributed to their ability to deal effectively with short text (e.g., tweets, user reviews, YouTube comments, etc.) [90].
- **Training settings:** to train our classifiers, we used 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 and 30% testing set).
- **Text pre-processing:** English stop-words were removed and stemming was applied to reduce words to their morphological roots. We used Weka's built-in stemmer and stop-word list to pre-process the posts in our dataset [57].

It is important to point out that lemmatization is sometimes used instead of stemming in app review classification tasks [73]. The results often show a marginal impact of these techniques on the precision of classification. In our analysis, we use stemming for its lower overhead. Specifically, lemmatization techniques are often exponential to the text length, while stemming is known for its linear time complexity [10].

- **Sentiment Analysis:** sentiment analysis is often used in app user feedback classification tasks as a classification feature of the input data [93, 51]. The underlying hypothesis is that user concerns are often expressed using negative sentiment [56]. To calculate the sentiment of our data, we used SentiStrength [85]. SentiStrength assigns positive (p) and negative (n) sentiment scores to input text, using a scale of -5 to +5, based on the emotional polarity of individual words. To convert SentiStrength's numeric scores into these categories, we adapted the approach proposed by Jongeling et al. [53] and Thelwall et al. [84]. Specifically, a post is considered positive if $p + n > 0$, negative if $p + n < 0$, and neutral if $p + n = 0$. It is worth mentioning that other sentiment analysis techniques, such as VADER and the Stanford CoreNLP are also used in related studies. However, the difference in performance between these tools is often marginal [73, 52, 92].
- **Text representation:** to classify our data, we experimented with simple *bag-of-words* with lowercase tokens. The *bag-of-words* representation encodes each post as a vector. Each attribute of the vector corresponds to one word in the vocabulary of the dataset. A word is included in the vocabulary if it is present in at least two posts. Words that appear in a single post are highly unlikely to carry any predictive value to the classifier. An attribute of one in a post's vector indicates that the corresponding word is present, while a zero indicates absence. This representation can be extended to treat common sequences of adjacent words, called n-grams, a gram is a single word; n is the number of adjacent words, so two adjacent words are a bi-gram. For example, the phrase *"this app is good"* contains four words, and three b-grams (*"this app", "app is", "is good"*). Fig. 4 illustrates how bag-of-words and n-gram representations work; *"updated"*, *"app"*, and *"crashes"* are the key words that occur in the tweet *"I updated the app, but now it crashes"*. *"Now it crashes"* is a tri-gram that is also included. Each '1' in the vector representation at the bottom corresponds to one of the highlighted n-grams, while each '0' corresponds to a vocabulary word that is not found in the tweet. To generate this representation, we utilized the n-gram tokenizer in Weka, which allowed uni-gram, bi-gram, and tri-gram tokens to be included in a single dataset.

We trained two set of classifiers to categorize our data. One classifier for detecting business posts and one classifier for detecting technical posts. The standard measures of Precision (P), Recall (R), and F-Score (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

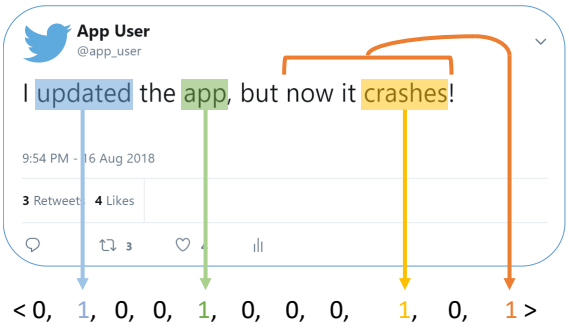


Fig. 4: A visual representation of an N-Gram encoded tweet.

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 $\beta = 2$ to emphasize recall over precision [8].

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[5]. Table 6 shows the performance of NB, SVM, and RF in terms of P, R, and F_2. SVM provided the best average classification performance in separating the different types of concerns, in comparison to NB and RF respectively. The best SVM results were obtained using the Pearson VII function-based universal kernel (Puk) with kernel parameters $\sigma = 8$ and $\omega = 1$ [88]. Universal Kernels are known to be effective for a large class of classification problems, especially for noisy data [79]. RF was evaluated with 100 iterations. Raising iterations above this number did not improve the performance. We also notice that almost all classifiers achieved better performance when classifying the reviews and tweets into generic categories of Business and Technical. The performance deteriorated when the data was classified at a subcategory level (Human, Market, Bug, and Feature) due to the fact that the classifier had to deal with a larger set of classes (labels). Separating concerns at this level can be challenging, especially when the data is relatively unbalanced.

In general, business-related posts were easier to classify than technically-related posts. This phenomenon is driven by the quantity of each class. Table 3 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 feature requests only occurring in 6.5% of posts. The relative sparsity of technical posts in comparison to other application domains can be explained based on the fact that the domain food delivery is a business domain in nature, thus, users had so many more business-related issues to discuss. For instance, Food courier services would often fail behind the scene, causing drivers to be dispatched to incorrect locations, or customer support

[5] A replication package is available at: `http://seel.cse.lsu.edu/data/ASEJ2019.zip`

Table 6: A comparison of the performance of our 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	0.85	0.79	0.82	0.89	0.85	0.87	0.85	0.87	0.86
LC + SEN	0.85	0.79	0.82	0.89	0.85	0.87	0.86	0.87	0.86
LC + SW	0.83	0.84	0.83	0.89	0.85	0.87	0.88	0.86	0.87
LC + SW + ST	0.84	0.83	0.84	0.89	0.85	0.87	0.87	0.88	0.87
Human									
LC	0.68	0.79	0.73	0.83	0.79	0.81	0.85	0.70	0.77
LC + SEN	0.68	0.78	0.73	0.83	0.79	0.81	0.85	0.69	0.76
LC + SW	0.69	0.83	0.75	0.83	0.79	0.81	0.87	0.74	0.80
LC + SW + ST	0.68	0.81	0.74	0.83	0.79	0.81	0.86	0.75	0.80
Market									
LC	0.56	0.69	0.62	0.72	0.66	0.69	0.78	0.47	0.59
LC + SEN	0.56	0.69	0.62	0.73	0.67	0.70	0.80	0.43	0.56
LC + SW	0.55	0.75	0.63	0.75	0.67	0.71	0.81	0.54	0.65
LC + SW + ST	0.56	0.74	0.64	0.75	0.68	0.71	0.79	0.53	0.64
Technical									
LC	0.38	0.67	0.49	0.60	0.55	0.57	0.93	0.07	0.13
LC + SEN	0.38	0.67	0.49	0.59	0.55	0.57	0.95	0.05	0.10
LC + SW	0.42	0.69	0.52	0.58	0.52	0.55	0.88	0.22	0.35
LC + SW + ST	0.39	0.71	0.51	0.61	0.55	0.58	0.91	0.16	0.27
Bugs									
LC	0.31	0.65	0.42	0.51	0.53	0.53	1.00	0.03	0.05
LC + SEN	0.31	0.65	0.42	0.51	0.53	0.51	0.52	0.03	0.06
LC + SW	0.34	0.66	0.45	0.52	0.57	0.53	0.91	0.14	0.24
LC + SW + ST	0.33	0.68	0.45	0.53	0.57	0.54	0.98	0.11	0.19
Features									
LC	0.17	0.65	0.27	0.50	0.51	0.51	0.00	0.00	0.00
LC + SEN	0.17	0.63	0.27	0.50	0.51	0.50	0.00	0.00	0.00
LC + SW	0.19	0.62	0.29	0.50	0.52	0.50	1.00	0.01	0.03
LC + SW + ST	0.18	0.67	0.28	0.51	0.52	0.51	1.00	0.01	0.02

to fail to call. These failures often caused customers to discuss competition and pricing. As a result, business concerns *crowded out* technical concerns. In other domains, failures are more immediate and visible to consumers, meaning that user concerns are more likely to take the form of bug reports.

We further experimented with the bag-of-words representation of text, and then allowing bi- and tri-grams to be included alongside individual words. Neither approach improved the performance. Table 7 shows a comparison between the uni-gram encoding (i.e., bag-of-words), and the encoding which included bi- and tri-grams. The lack of improvement partly stems from the fact that the additional composite tokens often had the same class implications as their constituent words. For example, the term *account* was found to have a negative implication on the *business* class, meaning that posts containing the word *account* were unlikely to be business-related. Most of the related N-grams, including *account got hacked* and *account was hacked* had the same

Table 7: A performance comparison of SVM using ordinary bag-of-words vs. N-grams.

	Uni, Bi, and Tri-Grams			**Bag-of-words**		
	Prec.	**Recall**	F_2	**Prec.**	**Recall**	F_2
Business	0.89	0.85	0.87	0.89	0.85	0.87
Human	0.84	0.80	0.82	0.83	0.79	0.81
Market	0.71	0.66	0.69	0.78	0.68	0.71
Technical	0.61	0.55	0.58	0.61	0.55	0.58
Bug	0.58	0.52	0.55	0.53	0.57	0.54
Feature	0.48	0.32	0.38	0.51	0.52	0.51

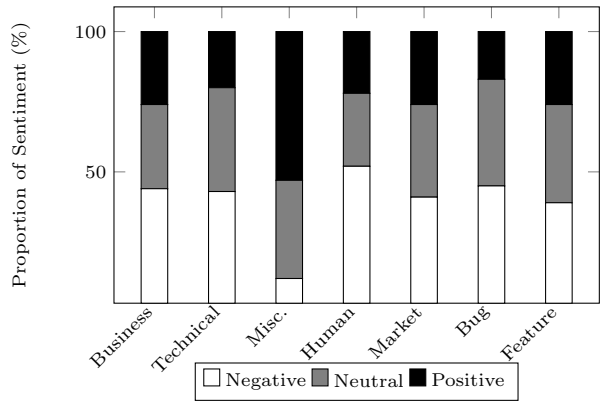


Fig. 5: The distribution of sentiment over the different types of posts.

implication, except with a substantially smaller weight. Therefore, they were essentially irrelevant to classification. In some other cases, bi- and tri-grams did not have the same implication as their constituent words. For example, *promo* was positively implicated to *business*, but *promo code* had a negative implication. However, the single word in this case, and in many others, had a higher weight than the bi- and tri-grams, and occurred in substantially more posts. Often times, the bi-grams had the same weight and occurrence as the tri-grams, making the tri-grams superfluous.

Our results also show that the sentiment polarity of posts had almost no impact on the classification accuracy. Specifically, the results show that miscellaneous posts (posts not business or technically-relevant) were detected as having more positive sentiment than any other category. These result were expected; non-miscellaneous posts often described problems users were having. Otherwise, as Fig. 5 shows, the categories had substantially similar sentiment scores overall. For future work, we suspect that enhancing SentiStrength's dictionary with emotion-provoking software-related words (crash, uninstall, etc.), or using customized sentiment analysis classifiers (e.g., [91]) would help to better estimate the emotional polarity of posts.

4.2.2 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 [1,30]. 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) [10] 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 [48]. Formally, $TF.IDF$ can be computed as:

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

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. In general, TF.IDF balances general frequency and appearance in number of posts. High frequent words appearing in few documents have higher TF.IDF. After defining $TF.IDF$, we extract important POS from the set of informative business and technical posts. The top ten nouns, verbs, and adjectives in our dataset are shown in Table 8.

4.2.3 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 a relation connecting these two entities. To count for such information, we use Pointwise Mutual Information (PMI).

Table 8: The top 10 most important nouns, verbs, and adjectives in our dataset.

Noun	**Verb**	**Adjective**
food	order	good
app	use	great
service	say	terrible
delivery	deliver	horrible
time	charge	wrong
consumer	wait	last
driver	cancel	bad
restaurant	give	free
money	want	easy

PMI is an information-theoretic measure of information overlap, or statistical dependence, between two words [22]. PMI was introduced by Church and Hanks [22], and later used by Turney [86] 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 corpus contains N documents, PMI between two words w_1 and w_2 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 $P(w_1, w_2)$ differs from the expectation based on the assumption of independence of $P(w_1)$ and $P(w_2)$ [15]. 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 probability 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 [65,69]. These attributes have made it an appealing similarity method to be used to process massive corpora of textual data in tasks such as short-text retrieval [65], Semantic Web [78,86], source code retrieval [55].

To generate the relations in our model, we computed PMI between every pair of words to determine their relatedness. One potential pitfall of relying on PMI as a measure of relatedness is that PMIs hits a maximum with words occurring only once. 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).

4.2.4 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. 6. 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 *late*.
- Triggers: on the left side of the node, we attach the list the top three verbs frequently associated (based on PMI) with the noun (concern's name). Verbs often represent *triggers*, or leading causes of concerns. In our example, the verbs *arrive*, *deliver*, and *prepare* 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 the parts of speech of a word, or $pos(word)$, $Adjs$, $Verbs$, 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 word w. These sets are defined, using the function $top(n, pred)$ to retrieve the top n words after words are sorted based on the predecessor function $pred(word)$. We use two functions to sort words: TF.IDF for nouns and PMI for verbs and adjectives. We express this using λ notation for defining anonymous functions, such that, $\lambda x.TFIDF(x)$ means define a function that takes an x and returns its TF.IDF. This results in the following expressions:

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

Fig. 6: The key elements of the entity-action-property relations represented by our model.

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

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

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. 7.

4.3 Model Evaluation and Interpretation

Due to the lack of *a priori* ground-truth, evaluating domain models 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 its ability to encompass the main concepts present in the knowledge it models [67, 76]. These concepts are often determined manually by domain experts. To evaluate our model generation procedure, we examine the main concepts captured in the model. Specifically, we assess the extent to which the noun-verb-adjective associations presented in our model reflect the main concerns identified by our qualitative analysis:

- **Customer Service**: Concerns about customer service frequently appeared when an *order* was not delivered on time, when the order was inaccurate, or when *refunds* were denied. The model identified both *customer* and *service* as important nouns along with the relations <*customer, refund*> and <*customer, incorrect*>. Furthermore, both *customer* and *service* were associated with the adjectives *poor* and *terrible* in the model.
- **Orders**: Orders were commonly associated with delays. Users complained about receiving *cold* food as a result. Users were *disappointed* whenever food was left waiting at the restaurant to be picked up. The model identified <*order, refuse*> whenever restaurants refused to cancel orders or the app refused to take action when things went wrong. In addition, <*order, place*> was a common occurrence as these two words often appeared together (e.g., *"place order"*). The relation <*order, second*> originated from posts of users

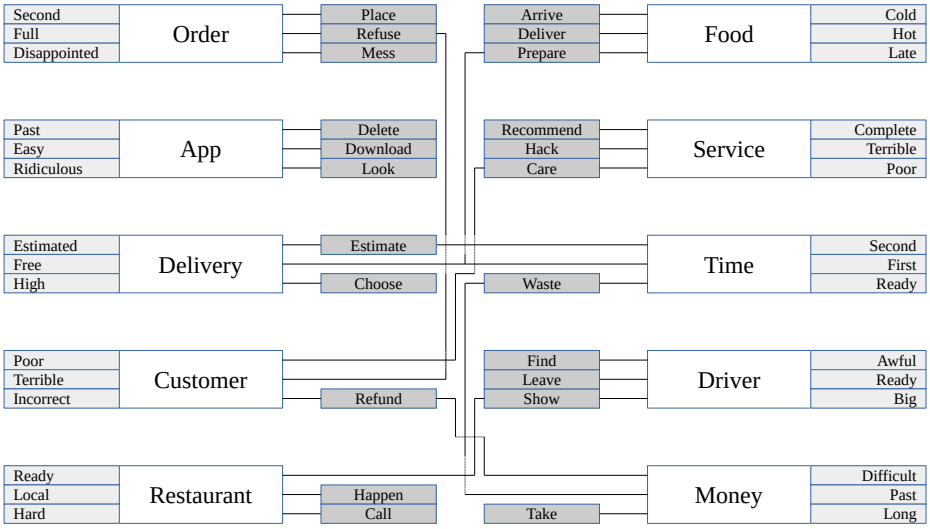


Fig. 7: A suggested model diagram depicting the relationships between important nouns (entities of the ecosystem), adjectives (attributes), and verbs (concern triggers).

complaining about having to re-order for the *second* time and the relation <*order, full*> originated from people asking for *full* refunds.

- **Food**: Food was directly related to arrival. This was captured in the relations <*food, arrive*>, <*food, deliver*>, and <*food, prepare*>. Food was also associated with temperature, mainly due to the number of complaints about receiving cold or hot food (e.g., <*food, cold*> and <*food, hot*>). Complaints about orders being late were common, resulting in the relation <*food, late*>.
- **Delivery**: Delivery was associated with a number of complaints about incorrect estimated times, explaining the relation <*delivery, estimate*>. 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 present in complaints about delivery delays. The relations <*time, estimate*> and <*time, prepare*> appeared for the same reasons they appeared with *delivery*. A common occurrence was <*time, waste*> due to unexpected delays and order cancellations. The relation <*time, long*> occurred in similar contexts, as in *"it took longer than the estimated time"*.

- **App**: App appeared alongside comments about ease-of-use, resulting in the relation <*app, easy*>. The relation <*app, ridiculous*> was a general complaint about poor policies or bad usability. The relation <*app, delete*> appeared when users discussed deleting an app after a poor experience. A common association was <*app, look*>, appearing due to phrases such as *"look into this"* and *"looks like"*. The relation <*app, end*> appeared from posts were users complained that they *"ended up"* eating cold or incorrect food, or not eating at all.
- **Money**: Money issues were captured by the relation <*money, waste*>. This relation stems from incidents were users ordered food that ended up being inedible and being unable to obtain a refund, which also yielded the model relation <*money, refund*>. The verb *take* was associated with money in posts such as *"you take my money but did not deliver"*, resulting the relation <*money, take*>.
- **Drivers**: Drivers are a critical component of the ecosystem. All services struggled with their drivers' timing, directions, and friendliness. Users frequently complained about drivers combining orders. The model successfully identified the relation <*driver, find*> from posts discussing a driver's inability to find their destination. Lack of friendliness is captured in the relation <*driver, awful*>.
- **Restaurants**: Users often asked services to add new restaurants as well as discussed problems that occurred between the app, restaurant, and driver. The relation <*restaurant, show*> appeared in the model partly due to users stating that the restaurant they wanted did not *"show up"* in the app. However, this phrase was more often associated with the *driver* not appearing at the restaurant. Communication problems between restaurants and consumers were captured through the <*restaurant, call*> relation.

In summary, to answer **RQ$_2$**, in terms of completeness (the omission of domain concepts and relationships), our model was able to recover a large number of concepts in the data. Missed concerns were rare (e.g., inability to find a customer service number). In terms of clarity, some of the captured relations, such as <*food, cold*> or <*money, waste*> were more obvious than others, for example <*restaurant, show*>. Incorrect, or hard to explain, relations were also present in the model. For example, the relations <*driver, big*> and <*money, long*> did not seem to reflect any issues that were identified by our qualitative analysis of the data, rather they originated from posts such as *"not a big fan of the driver"* or *"no longer interested"*. While these relations were relatively rare, they can be eliminated by compiling a list of such common English adjectives to filter them out before they make their way to the model. 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.

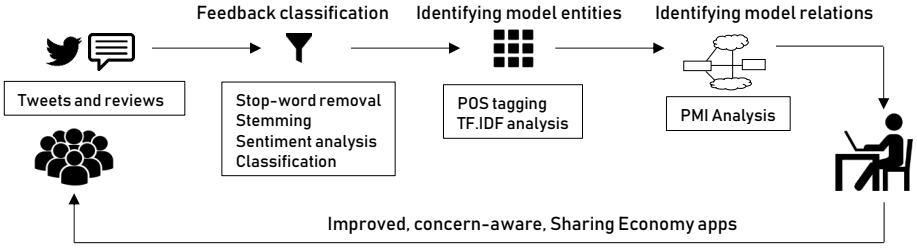


Fig. 8: A diagram of the proposed approach.

5 Discussion and Impact

A summary of the main steps of the proposed approach is depicted in Fig 8. The first phase of our analysis has revealed that user concerns in SE 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 rather than studied individually. In fact, such clusters can be automatically generated using app classification techniques [4].

Once these fine-grained categories of semantically-similar apps are identified, automated data clustering, classification, and modeling techniques should be employed to consolidate and analyze user feedback and 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. For instance, in the domain of food delivery apps, users preferred to use Twitter as low latency method to get instant reactions from app developers or operators. These complaints were very common whenever any of the services in our ecosystem went down for some reason. Understanding how users utilize different sources of feedback can help developers to focus their attention on the right channels of feedback while planning for their next release.

In the second phase of our analysis, we proposed an automated procedure for generating conceptual models of user concerns in the ecosystem of food delivery apps. According to Eric Yu [97], *"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"*. 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 [12], which requires large amounts of data and a calibration of several hyperparameters in order to produce meaningful topics [19, 40].

In terms of impact, our generated model can provide valuable ecosystem-wide information to SE app developers, acting as a vehicle to facilitate a quick transition from domain knowledge to requirements specifications. For instance, startups, or newcomers, trying to break into the food delivery app market, can use our procedure to quickly generate a model for their micro-ecosystem of operation. Through the model entities and relations, they can get insights into the complex realities of their operational environments. Such information can help them to redirect their effort toward innovations that can help to avoid these issues in their apps. For example, 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.

6 Limitations and Validity

Our analysis takes the form of a case study. Case studies often suffer from external validity threats since they target specific phenomena in their specific contexts [94]. 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 *fittest* actors in the ecosystem. These popular apps often receive significantly more feedback than smaller apps [63]. Furthermore, to minimize any sampling bias, our data collection process included multiple sources of user feedback and has extended over a long period of time to capture as much information about the apps in our ecosystem as possible. In terms of generalizability, we anticipate that our proposed approach could be applied to other application domains beyond SE, especially for apps operating in complex multi-agent ecosystems. However, independent case studies need to be conducted before we can make such a claim.

Internal validity threats may stem from the fact that we only relied on the textual content of user posts 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 [40]. 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 [40].

Threats might also stem from our model evaluation procedure. Specifically, our generated LDA topics and models 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.

7 Conclusions

SE has come with a set of unconventional challenges for software engineers. Understanding these challenges begins with understanding end-users' needs, and then using such knowledge to develop a better understanding of the internal dynamics of such a complex and dynamic software ecosystem. To achieve this goal, in this paper, we proposed an automated approach for modeling crowed feedback in ecosystems of SE apps. 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 often extend over a broad range of technical and business issues. The results also showed that, in our ecosystem of interest, business concerns were more prevalent than technical concerns. In the second phase of our analysis, we proposed an approach for automatically generating an abstract conceptual model of the main user concerns in the ecosystem of food delivery apps. The results showed that a descriptive model can 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. We further compared our generated model's entities with topics generated using the topic modeling technique LDA. The results showed that, due to the short nature and lack of structure in user feedback text, LDA failed to generate any cohesive topics that were representative of valid user concerns.

In addition to extrinsically evaluating our generated model, our future work in this domain will include conducting more case studies, targeting SE apps operating in dynamic and multi-agent ecosystems, such as *ridesharing* or *freelancing*. 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 SE app market at a micro level and provide more succinct representations of its complex realities.

Acknowledgements This work was supported in part by the U.S. National Science Foundation (Award CNS 1951411) and LSU Economic Development Assistantships awards.

References

1. Abbott, R.: Program design by informal English descriptions. Communications of the ACM **26**(11), 882–894 (1983)
2. Acquier, A., Daudigeos, T., Pinkse, J.: Promises and paradoxes of the sharing economy: an organizing framework. Technological Forecasting and Social Change **125**, 1–10 (2017)
3. Ala-Mantilaa, S., Ottelina, J., Heinonenb, J., Junnilaa, S.: To each their own? The greenhouse gas impacts of intra-household sharing in different urban zones. Journal of Cleaner Production **135**, 356–367 (2016)
4. AlSubaihin, A., Sarro, F., Black, S., Capra, L., Harman, M., Jia, Y., Zhang, Y.: Clustering mobile apps based on mined textual features. In: International Symposium on Empirical Software Engineering and Measurement, pp. 38:1–38:10 (2016)
5. Aznar, J., Sayeras, J.M., Rocafort, A., Galiana, J.: The irruption of Airbnb and its effects on hotels' profitability: An analysis of barcelona's hotel sector. Intangible Capital **13**(1), 147–159 (2017)
6. Bellotti, V., Ambard, A., Turner, D., Gossmann, C., Demkova, K., Carroll., J.M.: A muddle of models of motivation for using peer-to-peer economy systems. In: Annual ACM Conference on Human Factors in Computing Systems, pp. 1085–1094 (2015)
7. Benkler, Y.: Peer production, the commons, and the future of the firm. Strategic Organization **15**(2), 264–274 (2017)
8. Berry, D.: Evaluation of tools for hairy requirements and software engineering tasks. In: International Requirements Engineering Conference Workshops, pp. 284–291 (2017)
9. Bing, L., Lam, W., Wong, T.L.: Using query log and social tagging to refine queries based on latent topics. In: International Conference on Information and Knowledge Management, pp. 583–592 (2011)
10. Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python. O'Reilly Media (2009)
11. Bistaffa, F., Farinelli, A., Chalkiadakis, G., Ramchurn, S.: Payments for large-scale social ridesharing. In: The ACM Conference on Recommender Systems, pp. 139–146 (2015)
12. Blei, D., Ng, A., Jordan, M.: Latent Dirichlet Allocation. Journal of Machine Learning Research **3**, 993–1022 (2003)
13. Bonciu, F., Balgar, A.: Sharing economy as a contributor to sustainable growth. An EU perspective. Romanian Journal of European Affairs **16**(2), 36–45 (2016)
14. Bond, A.: An app for that: local governments and the rise of the sharing economy. Notre Dame Law Review (Online 77) (2014)
15. Bouma, G.: Normalized (pointwise) mutual information in collocation extraction. German Society for Computation Linguistics and Language Technology pp. 31–40 (2009)
16. Cannon, S., Summers, L.: How Uber and the sharing economy can win over regulators. Harvard Business Review **13** (2014)

17. Carreńo, G., Winbladh, K.: Analysis of user comments: An approach for software requirements evolution. In: International Conference on Software Engineering, pp. 343–348 (2013)
18. Chen, G., Cheng, M., Edwards, D., Xu, L.: Covid-19 pandemic exposes the vulnerability of the sharing economy. Research Square (2020)
19. Chen, N., Lin, J., Hoi, S., Xiao, X., Zhang, B.: AR-Miner: Mining informative reviews for developers from mobile app marketplace. In: International Conference on Software Engineering, pp. 767–778 (2014)
20. Cheng, M.: Current sharing economy media discourse in tourism. Annals of Tourism Research **60**(C), 111–114 (2016)
21. Chow, Y., Yuan Yu, J.: Real-time bidding based vehicle sharing. In: International Conference on Autonomous Agents and Multiagent Systems, pp. 1829—-1830 (2015)
22. Church, K., Hanks, P.: Word association norms, mutual information, and lexicography. Computer Linguistics **16**(1), 22–29 (1990)
23. Cleland-Huang, J., Settimi, R., BenKhadra, O., Berezhanskaya, E., Christina, S.: Goal-centric traceability for managing non-functional requirements. In: International Conference on Software Engineering, pp. 362–371 (2005)
24. Conger, K., Griffith, E.: The results are in for the sharing economy. they are ugly. (2020). URL `https://www.nytimes.com/2020/05/07/technology/the-results-are-in-for-the-sharing-economy-they-are-ugly.html`
25. Coulton, P., Bamford, W.: Experimenting through mobile apps and app stores. International Journal on Mobile Human Computer Interaction **3**(4), 55–70 (2011)
26. Dillahunt, T., Malone, A.: The promise of the sharing economy among disadvantaged communities. In: Annual ACM Conference on Human Factors in Computing Systems, pp. 2285—-2294 (2015)
27. Dillahunt, T., Wang, X., Wheeler, E., Cheng, H.F., Hecht, B., Zhu, H.: The sharing economy in computing: A systematic literature review. ACM Human Computer Interaction **1**(38), 1–26 (2017)
28. Dogru, T., Mody, M., Suess, C.: Adding evidence to the debate: Quantifying Airbnb's disruptive impact on ten key hotel markets. Tourism Management **72**, 27–39 (2019)
29. Edelman, B., Luca, M.: Digital discrimination: The case of Airbnb.com. Harvard Business School NOM Unit Working Paper No. 14-054 (2014)
30. Elbendak, M., Vickers, P., Rossiter, N.: Parsed use case descriptions as a basis for object-oriented class model generation. Journal of Systems and Software **87**(7), 1209–1223 (2011)
31. Finkelstein, A., Harman, M., Jia, Y., Martin, W., Sarro, F., Zhang, Y.: App store analysis: Mining app stores for relationships between customer, business and technical characteristics. Tech. rep., University of College London, Tech. Rep. rN/14/10, 2014 (2014)
32. Ge, Y., Knittel, C., MacKenzie, D., Zoepf, S.: Racial and gender discrimination in transportation network companies (NBER Working Paper No. 22776) (2017)
33. Glassey, O.: Method and instruments for modeling integrated knowledge. Knowledge and Process Management **15**(4), 247–257 (2008)
34. Glinz, M.: On non-functional requirements. In: Requirements Engineering, pp. 21–26 (2007)
35. Goel, P., Kulik, L., Ramamohanarao, K.: Privacy-aware dynamic ride sharing. ACM Transactions on Spatial Algorithms and Systems **2**(1), 1–41 (2016)
36. Gomez, M., Martineza, M., Monperrus, M., Rouvoy, R.: When app stores listen to the crowd to fight bugs in the wild. In: International Conference on Software Engineering, track on New Ideas and Emerging Results (NIER), vol. 2, pp. 567–570 (2015)
37. Griffiths, T., Steyvers, M.: Finding scientific topics. In: The National Academy of Sciences, pp. 5228–5235 (2004)
38. Groen, E., Kopczyǹska, S., Hauer, M., Krafft, T., Doerr, J.: Users: The hidden software product quality experts?: A study on how app users report quality aspects in online reviews. In: International Requirements Engineering Conference, pp. 80–89 (2017)
39. Guzman, E., Ibrahim, M., Glinz, M.: A little bird told me: Mining tweets for requirements and software evolution. In: International Requirements Engineering Conference, pp. 11–20 (2017)

40. Guzman, E., Maalej, W.: How do users like this feature? A fine grained sentiment analysis of app reviews. In: Requirements Engineering, pp. 153–162 (2014)
41. Harman, M., Jia, Y., Zhang, Y.: App store mining and analysis: MSR for app stores. In: Conference on Mining Software Repositories, p. 2012 (108–111)
42. He, W., Li, D., Zhang, T., An, L., Guo, M., Chen, G.: Mining regular routes from GPS data for ridesharing recommendations. In: The ACM SIGKDD International Workshop on Urban Computing, pp. 79–86 (2012)
43. Hofmann, T.: Probabilistic latent semantic indexing. In: International Conference on Research and Development in Information Retrieval, pp. 50–57 (1999)
44. Hong, L., Davison, B.: Empirical study of topic modeling in Twitter. In: Workshop on Social Media Analytics, pp. 80–88 (2010)
45. Hossain, M.: Sharing economy: A comprehensive literature review. International Journal of Hospitality Management **87** (2020)
46. Hüttel, A., Ziesemer, F., Peyer, M., Balderjahn, I.: To purchase or not? Why consumers make economically (non-) sustainable consumption choices. Journal of Cleaner Production **174**, 827–836 (2018)
47. Iacob, C., Harrison, R.: Retrieving and analyzing mobile apps feature requests from online reviews. In: Mining Software Repositories, pp. 41–44 (2013)
48. Inouye, D., Kalita, J.: 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), pp. 298–306 (2011)
49. Jansen, S., Finkelstein, A., Brinkkemper, S.: A sense of community: A research agenda for software ecosystems. In: International Conference on Software Engineering - Companion Volume, pp. 187–190 (2009)
50. Jha, N., Mahmoud, A.: Using frame semantics for classifying and summarizing application store reviews. Empirical Software Engineering **23**(6), 3734–3767 (2018)
51. Jha, N., Mahmoud, A.: Mining non-functional requirements from app store reviews. Empirical Software Engineering (2019)
52. Jongeling, R., Datta, S., Serebrenik, A.: Choosing your weapons: On sentiment analysis tools for software engineering research. In: International Conference on Software Maintenance and Evolution, pp. 531–535 (2015)
53. Jongeling, R., Sarkar, P., Datta, S., Serebrenik, A.: On negative results when using sentiment analysis tools for software engineering research. Empirical Software Engineering **22**(5), 2543–2584 (2017)
54. Khalid, H., Shihab, E., Nagappan, M., Hassan, A.: What do mobile app users complain about? IEEE Software **32**(3), 70–77 (2015)
55. Khatiwada, S., Tushev, M., Mahmoud, A.: Just enough semantics: An information theoretic approach for IR-based software bug localization. Information and Software Technology **93**, 45–57 (2017)
56. Lin, B., Zampetti, F., Bavota, G., Di Penta, M., Lanza, M., Oliveto, R.: Sentiment analysis for software engineering: How far can we go? In: International Conference on Software Engineering (2018)
57. Lovins, J.: Development of a stemming algorithm. Mechanical Translation and Computational Linguistics **11**, 22–31 (1968)
58. Maalej, W., Nabil, H.: Bug report, feature request, or simply praise? On automatically classifying app reviews. In: Requirements Engineering Conference, pp. 116–125 (2015)
59. Martin, C.: The sharing economy: A pathway to sustainability or a nightmarish form of neoliberal capitalism? Ecological Economics **121**, 149–159 (2016)
60. Martin, W., Harman, M., Jia, Y., Sarro, F., Zhang, Y.: The app sampling problem for app store mining. In: Working Conference on Mining Software Repositories, pp. 123–133 (2015)
61. Martin, W., Sarro, F., Jia, Y., Zhang, Y., Harman, M.: A survey of app store analysis for software engineering. Transactions on Software Engineering **43**(9), 817–847 (2017)
62. Matzler, K., Veider, V., Kathan, W.: Adapting to the sharing economy. MIT Sloan Management Review **56**(2), 71–77 (2015)
63. Mcilroy, S., Shang, W., Ali, N., Hassan, A.: User reviews of top mobile apps in apple and google app stores. Communications of the ACM **60**(11), 62–67 (2017)

64. Mcllroy, S., Ali, N., Khalid, H., Hassan, A.: Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. Empirical Software Engineering **21**(3), 1067–1106 (2016)
65. Mihalcea, R., Corley, C., Strapparava, C.: Corpus-based and knowledge-based measures of text semantic similarity. In: National Conference on Artificial Intelligence, pp. 775–780 (2006)
66. Mimno, D., Wallach, H., Talley, E., Leenders, M., McCallum, A.: Optimizing semantic coherence in topic models. In: Conference on Empirical Methods in Natural Language Processing, pp. 262–272 (2011)
67. Mohagheghi, P., Dehlen, V.: Existing model metrics and relations to model quality. In: ICSE Workshop on Software Quality, pp. 39–45 (2019)
68. Murillo, D., Buckland, H., Val, E.: When the sharing economy becomes neoliberalism on steroids: unravelling the controversies. Technological Forecasting and Social Change **125**, 66–76 (2017)
69. Newman, D., Noh, Y., Talley, E., Karimi, S., Baldwin, T.: Evaluating topic models for digital libraries. In: Annual Joint Conference on Digital Libraries, pp. 215–224 (2010)
70. Nonaka, I.: A dynamic theory of organizational knowledge creation. Organization Science **5**(1), 14–37 (1994)
71. Pagano, D., Maalej, W.: User feedback in the appstore: An empirical study. In: Requirements Engineering Conference, pp. 125–134 (2013)
72. Palomba, F., Linares-Vásquez, M., Bavota, G., Oliveto, R., Di Penta, M., Poshyvanyk, D., De Lucia, A.: Crowdsourcing user reviews to support the evolution of mobile apps. Journal of Systems and Software **137**, 143–162 (2018)
73. Panichella, S., Di Sorbo, A., Guzman, E., Visaggio, C., Canfora, G., Gall, H.: How can I improve my app? Classifying user reviews for software maintenance and evolution. In: International Conference on Software Maintenance and Evolution, pp. 767–778 (2015)
74. PwC: The sharing economy: Consumer intelligence series. PricewaterhouseCoopers LLP (2015)
75. Rehurek, R., Sojka, P.: Software framework for topic modelling with large corpora. In: Workshop on New Challenges for NLP Frameworks (2010)
76. Rubén: Domain analysis: An introduction
77. Sorbo, A.D., Panichella, S., Alexandru, C., Shimagaki, J., Visaggio, C., Canfora, G., Gall, H.: What would users change in my app? Summarizing app reviews for recommending software changes. In: International Symposium on Foundations of Software Engineering, pp. 499–510 (2016)
78. Sousa, D., Sarmento, L., Rodrigues, E.: Characterization of the twitter replies network: Are user ties social or topical? In: International Workshop on Search and Mining User-generated Contents, pp. 63–70 (2010)
79. Steinwart, I.: On the influence of the kernel on the consistency of Support Vector Machines. Journal of Machine Learning Research **2**, 67–93 (2001)
80. Svedic, Z.: The effect of informational signals on mobile apps sales ranks across the globe. Ph.D. thesis, School Business Faculty, Simon Fraser University (2015)
81. Tang, J., Meng, Z., Nguyen, X., Mei, Q., Zhang, M.: Understanding the limiting factors of topic modeling via posterior contraction analysis. In: International Conference on Machine Learning, pp. 190–198 (2014)
82. Thebault-Spieker, J., Terveen, L., Hecht, B.: Avoiding the south side and the suburbs: The geography of mobile crowdsourcing markets. In: The ACM Conference on Computer Supported Cooperative Work & Social Computing, pp. 265–275 (2015)
83. Thebault-Spieker, J., Terveen, L., Hecht, B.: Toward a geographic understanding of the sharing economy: Systemic biases in uberx and taskrabbit. ACM Transactions on Computer-Human Interaction **24**(3), 40 (2017)
84. Thelwall, M., Buckley, K., Paltoglou, G.: Sentiment strength detection for the social web. Journal of the American Society for Information Science and Technology **63**(1), 163–173 (2012)
85. Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., Kappas, A.: Sentiment strength detection in short informal text. Journal of the Association for Information Science and Technology **61**(12), 2544–2558 (2010)
86. Turney, P.: Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In: European Conference on Machine Learning, pp. 491–502 (2001)

87. Tussyadiah, I.P., Pesonen, J.: Impacts of peer-to-peer accommodation use on travel patterns. Journal of Travel Research **55**(8), 1022–1040 (2016)
88. Üstün, B., Melssen, W., Buydens., L.: Facilitating the application of support vector regression by using a universal pearson vii function based kernel. Chemometrics and Intelligent Laboratory Systems **81**, 29–40 (2006)
89. Villarroel, L., Bavota, G., Russo, B., Oliveto, R., Di Penta, M.: Release planning of mobile apps based on user reviews. In: International Conference on Software Engineering, pp. 14–24 (2016)
90. Wang, S., Manning, C.: Baselines and bigrams: Simple, good sentiment and topic classification. In: Annual Meeting of the Association for Computational Linguistics, pp. 90–94 (2012)
91. Williams, G., Mahmoud, A.: Analyzing, classifying, and interpreting emotions in software users' tweets. In: International Workshop on Emotion Awareness in Software Engineering, pp. 2–7 (2017)
92. Williams, G., Mahmoud, A.: Mining Twitter feeds for software user requirements. In: International Requirements Engineering Conference, pp. 1–10 (2017)
93. Williams, G., Mahmoud, A.: Modeling user concerns in the app store: A case study on the rise and fall of Yik Yak. In: International Requirements Engineering Conference, pp. 64–75 (2018)
94. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslèn, A.: Experimentation in Software Engineering. Springer (2012)
95. Xu, L., Shah, N., Chen, L., Diallo, N., Gao, Z., Lu, Y., Shi, W.: Economy: Privacy respecting contract based on public blockchain. In: The ACM Workshop on Blockchain, Cryptocurrencies and Contracts, pp. 15–21 (2017)
96. Yan, X., Guo, J., Lan, Y., Cheng, X.: A biterm topic model for short texts. In: International Conference on World Wide Web, pp. 1445–1456 (2013)
97. Yu, E.S.: Conceptual modeling: Foundations and applications. chap. Social Modeling and i*, pp. 99–121. Springer-Verlag (2009)
98. Zervas, G., Proserpio, D., Byers, J.: The rise of the sharing economy: Estimating the impact of Airbnb on the hotel industry. Journal of Marketing Research **54**(5), 687–705 (2017)
99. Zhao, W., Jiang, J., Weng, J., Lim, E., Yan, H., Li, X.: Comparing Twitter and traditional media using topic models. In: European Conference on Advances in Information Retrieval, pp. 338–349 (2011)
100. Zhu, G., Kam Fung So, K., Hudson, S.: Inside the sharing economy: Understanding consumer motivations behind the adoption of mobile applications. International Journal of Contemporary Hospitality Management **29**(9), 2218–2239 (2017)