4-5-2023

Domain Specific Analysis of Privacy Practices and Concerns in the Mobile Application Market

Fahimeh Ebrahimi Meymand
Louisiana State University and Agricultural and Mechanical College

DOMAIN SPECIFIC ANALYSIS OF PRIVACY PRACTICES AND CONCERNS IN THE MOBILE APPLICATION MARKET

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

Division of Computer Science and Engineering

by
Fahimeh Ebrahimi Meymand
B.Sc., Information Technology Engineering, 2013
M.Sc., Information Technology Engineering, 2015
March 2023

This dissertation is dedicated to my loving husband, Ramin Riahipour. His unconditional love, unwavering support, and endless encouragement have made this journey truly unforgettable. I would like to thank him for being my rock, my partner, and my best friend. I am forever grateful for having him by my side.

Acknowledgments

I would like to thank my committee members, Dr. Gerald Baumgartner and Dr. Golden G Richard III, for their time and support during this journey. I would also like to thank Dr. Gabriele Piccoli for serving as a dean representative on my dissertation committee. A special thank you goes to my Academic Advisor, Dr. Anas (Nash) Mahmoud, for his enormous support and mentorship during the past four years. Finally, I would like to acknowledge National Science Foundation (NSF) and the LSU Economic Development Assistantships (EDA) for sponsoring this work.

Table of Contents

Abstract

Mobile applications (apps) constantly demand access to sensitive user information in exchange for more personalized services. These—mostly unjustified—data collection tactics have raised major privacy concerns among mobile app users. Existing research on mobile app privacy aims to identify these concerns, expose apps with malicious data collection practices, assess the quality of apps' privacy policies, and propose automated solutions for privacy leak detection and prevention. However, existing solutions are generic, frequently missing the contextual characteristics of different application domains. To address these limitations, in this dissertation, we study privacy in the app store at a domain level. Our objective is to propose automated solutions that are tailored to the specific data collection practices of each operational domain.

The analysis in this dissertation can be divided into three main phases. In the first phase, we propose an automated solution to classify apps in the mobile app market into more coherent categories of functionally-related apps. In the second phase, we propose an effective approach for summarizing users' privacy concerns in mobile app reviews. Our objective is to help app developers identify and understand the most critical privacy challenges in their specific domain of operation. In the third phase, we conduct a qualitative analysis of mobile apps' privacy nutrition labels. Our objectives are to explore the information value of such labels, quantify the discrepancies between apps' privacy policies and their data collection labels, and identify privacy outlier apps in each domain. Based on our analysis, we suggest several design strategies to help app stores preserve the credibility and utility of their privacy label systems.

Chapter 1. Introduction

Privacy can be hard to define. The word itself is derived from the Latin *Privatus*, which means *"withdraw from public life."* However, the majority of modern definitions can be traced back to Brandeis and Warren [1] who defined privacy in 1890 as *"the right to be let alone."* This definition provided a basis for ensuring the legal protection of privacy as a fundamental human right. Since then, this definition has been revisited numerous times to count for the plethora of political, social, and technological advances in society. For instance, in 1968, Westin [2] described privacy as *"the right to select what personal information about me is known to what people."* This definition emphasized privacy as a control over personal information. To further meet the growing dimensions of the concept, in 2002, Solove [3] proposed another conceptualizing of privacy, including conceptions of limited access to the self, secrecy, control over personal information, personhood, individuality, dignity, autonomy, antitotalitarianism, and intimacy.

The proliferation of mobile devices over the past decade along with their unique operational characteristics have imposed new challenges on end-users privacy. In general, mobile apps are designed with a set of user goals. A goal can be described as any desirable objective that the system under consideration should achieve [4]. However, driven by the fierce market competition, app developers often deviate from their original goals [5]. These deviations typically come in the form of extreme privacy-invading tactics, such as constant location-tracking [6], unsolicited data collection [5, 7], or any form of features that are engineered to lure users to trade their privacy for more personalized services [8, 9].

Parts of this chapter were previously published as Ebrahimi, Fahimeh, Miroslav Tushev, and Anas Mahmoud. "Mobile app privacy in software engineering research: A systematic mapping study." Information and Software Technology (2021).

These intrusive, and sometimes borderline unethical, practices have led to the emergence of new and more significant privacy concerns among mobile app users. Such concerns often revolve around the types of information apps are asking for, who should or should not have access to this information, and how to prevent misuse of this access. In fact, these new challenges have prompted researchers to broaden existing definitions to include, in addition to personal information, device-specific information that can be used as identifiers, including installed apps, connected WIFI, operating system's build information, and carrier [9].

1.1. Background and Literature Review

The research on mobile app privacy has witnessed a rapid growth over the past decade. In general, studies in this domain cover a broad range of topics, tackling privacy from a user, developer, and system perspectives. In our recent systematic review, we categorized existing evidence on mobile app privacy in the software engineering literature into four different categories [10]. These categories can be described as follows:

- **Privacy policy:** Mobile apps' privacy policies act as legally-binding contracts between app developers and users. App developers disclose in their policies how user information is being collected, used, shared, and protected by the app [7, 11]. Privacy policy research aims to detect policy violations by mapping the privacy claims in the app's privacy policy to the information flow in the app's code. Static code analysis tools, such as FlowDroid [12], are commonly used to detect possible data leaks in mobile apps. Recent studies in this domain have revealed that a large percentage of apps are either non-compliant with their stated privacy claims, do not

maintain a privacy policy, or provide a policy that is either ambiguous or incomprehensible to the average users [13, 7, 14].

- **Privacy in user feedback:** App stores provide a mechanism for app users to express their opinions about apps in the form of textual reviews and meta-data (e.g., star ratings). Recent research has revealed that privacy concerns, while not as common as other functional issues (e.g., bug reports and maintenance requests) are widespread among almost all categories of mobile apps, and are often accompanied with lower app ratings [15, 16]. These concerns are also often expressed using more varied language than other technical issues [15].

- **Privacy requirements analysis:** This category of research is focused on extracting, specifying, and enforcing privacy requirements of mobile apps [17, 18, 19, 20]. In particular, researchers in this domain frequently propose and evaluate frameworks, tools, heuristics, rules, and templates to enable structured analysis of the various privacy and security problems experienced by mobile app users and translate these problems into actionable software requirements [21, 22].

- **Privacy leak analysis:** Research under this category is focused on introducing effective methods for privacy leak detection and prevention. These methods often take the form of working tools that employ static, dynamic, or mixed models of code analysis to identify apps with malicious data collection behavior and detect components of mobile apps (e.g., code, API calls, GUI components) that might lead to information leakage [23, 24, 25, 26].

1.2. Contribution and Outline

The main thrust of this dissertation is the recognition that privacy in the mobile app market is a domain-specific problem. Each operational domain has its own privacy paradox; what is considered a normal privacy practice in one domain, may be considered an invasive, or even malicious practice, in another domain. Therefore, generic *one-size-fits-all* solutions, while can help to understand the grand challenges of privacy, often fail to address the unique contextual characteristics of each application domain. To address these limitations, this dissertation makes the following contributions:

- In **Chapter 2**, we propose an automated approach for classifying mobile apps into more focused groups of functionally-related application domains. Modern application stores enable developers to classify their apps by choosing from a set of generic categories, or genres, such as health, games, and music. These categories are typically static—new categories do not necessarily emerge over time to reflect innovations in the mobile software landscape. With thousands of apps classified under each category, locating apps that match a specific consumer interest or share the same core functionalities can be a challenging task. To overcome these limitations, we employ word embeddings to generate numeric semantic representations of app descriptions. These representations are then classified to generate more cohesive categories of apps. Such fine-grained categorization is intended to automatically and accurately identify the functional boundaries of app collections, thus enable an in-depth understanding of the operational characteristics of different application domains.

- In **Chapter 3**, we propose a domain-specific approach for summarizing user privacy concerns in mobile app reviews. Our analysis is conducted using a dataset of 2.6 million app reviews sampled from three different application domains. The results show that users in different application domains express their privacy concerns using domain-specific vocabulary. This vocabulary is then leveraged to help unsupervised automated text summarization algorithms to generate concise and comprehensive summaries of privacy concerns in app review collections. Our analysis is intended to help app developers quickly and accurately identify the most critical privacy concerns in their domain of operation, and ultimately, alter their data collection practices to address these concerns.

- In **Chapter 4**, we qualitatively analyze privacy labels of mobile across multiple application domains. App stores have recently rolled out privacy labels as a simplified and more structured view of the data types that apps may collect and link to their users' identity. Our analysis aims to study the privacy practices of mobile apps across and within different application domains through their privacy labels. Our objectives are to explore the information value of such labels, quantify the discrepancies between apps' privacy policies and their data collection labels, and identify privacy outlier apps in each domain. Based on our analysis, we suggest several design strategies to help app stores preserve the credibility and utility of their label systems. Our suggestions take the form of visual nudges that are designed to steer app users towards apps that preserve their privacy, and at the same time, prompt app developers to keep their data collection practices in-check.

Chapter 2. Mobile App Classification

Modern application stores enable developers to classify their apps by choosing from a set of generic categories, or genres, such as health, games, and music. These categories are typically static—new categories do not necessarily emerge over time to reflect innovations in the mobile software landscape. With thousands of apps classified under each category, locating apps that match a specific consumer interest can be a challenging task. To overcome this challenge, in this chapter, we propose an automated approach for classifying mobile apps into more focused categories of functionally-related application domains. Our aim is to enhance apps visibility and discoverability. Specifically, we employ word embeddings to generate numeric semantic representations of app descriptions. These representations are then classified to generate more cohesive categories of apps. Our empirical investigation is conducted using a dataset of 600 apps, sampled from the Education, Health&Fitness, and Medical categories of the Apple App Store. The results show that, our classification algorithms achieve their best performance when app descriptions are vectorized using GloVe, a count-based word embedding. Our findings are further validated using a dataset of Sharing Economy apps and the results are evaluated by 12 human subjects. The results show that GloVe combined with Support Vector Machines can produce app classifications that are aligned to a large extent with human-generated classifications.

2.1. Introduction

Over the past decade, mobile application (app) stores, such as Google Play and the Apple App Store, have expanded in size to host millions of apps, offering app users

This chapter was previously published as Ebrahimi, Fahimeh, Miroslav Tushev, and Anas Mahmoud. "Classifying mobile applications using word embeddings." ACM Transactions on Software Engineering and Methodology (TOSEM) 31.2 (2021): 1-30.

virtually unlimited options to choose from. These apps are typically classified under several categories (e.g., Gaming) and subcategories (e.g., Sport, Board, and Card) that are intended to help consumers discover apps more effectively. For instance, the Apple App Store, which currently hosts close to 1.8 million apps, classifies apps under 23 distinct categories, while Google Play, which currently hosts close to 2.87 million apps, offers 35 distinct categories of apps.

With thousands of apps classified under each category, locating apps that match a specific consumer interest can be a challenging task [27]. Furthermore, categorizing apps under broad categories of loosely related functionalities can severely impact their discoverability, thus their download rates and chances of survival. These challenges have encouraged experts, across a broad range of application domains, to propose more accessible classification schemes of apps in their fields [28, 29, 30]. For instance, apps under the Health&Fitness category are often classified by healthcare professionals into more specific categories (e.g., health interventions, consulting, and patient management, etc.) to increase their visibility to doctors and patients [29, 30]. However, these classifications are often static, relying on a manual synthesis of app descriptions or their usage scenarios. Therefore, they can hardly adapt to the rapid pace of innovation in the app market or the large number of new apps approved daily by popular app stores.

Automated approaches that are proposed to solve this problem often employ standard classification techniques to categorize apps based on their publicly available app store descriptions [31, 32, 33, 34]. Other information, such as download rates of apps, their usage scenarios, ratings, and source code have also been used to generate more accurate classification models [35, 36]. However, these techniques are often limited by the restricted

syntactic nature of app descriptions (e.g., text sparsity and vocabulary mismatch) [37], the complexity associated with collecting certain types of app information (e.g., source code and usage scenarios), and the general lack of expert-generated ground-truths to assess the performance of generated models.

To overcome these limitations, in this chapter, we propose an automated approach for classifying mobile apps using word embeddings. Word embeddings produce semantic vector representations of words in a text collection [38]. Such numeric representations can be used to estimate the semantic similarity between words in apps descriptions, thus, help to overcome the syntactic limitations of these descriptions [39]. Our empirical analysis is conducted using a dataset of 600 apps, sampled from the Education, Health&Fitness, and Medical app categories of the Apple App Store. Existing expert-generated classifications of apps are then used to assess the accuracy of our classifiers and compare their performance to several existing classification and text modeling methods [28, 29]. Our approach is then validated using a dataset of Sharing Economy apps, sampled from a broad range of application domains, such as ride-sharing (e.g. Uber and Lyft), lodging (e.g., Airbnb and Couchsurfing), and freelancing (e.g., TaskRabbit and UpWork). We further conduct a study with 12 participants (judges) to assess the quality of our generated classifications.

The remainder of this chapter is organized as follows. Section 2.2 reviews related work and motivates our research. Section 2.3 introduces word embeddings. Section 2.4 describes our research oracle. Section 2.5 presents our experimental setup and analysis. Section 2.6 describes our human study. Section 2.7 discusses our main findings. Section 2.8 describes the potential limitations of our study. Finally, Section 2.9 concludes the chapter and presents our directions of future work.

2.2. Related Work and Motivation

In this section, we review existing work related to mobile app classification, discuss its limitations, and motivate our approach.

2.2.1. Related work

Motivated by the vast growth of mobile app stores, the research on mobile apps classification has gained considerable momentum over the past decade. For instance, Zhu et al. [40, 35] proposed an automated approach for classifying mobile apps in the Nokia Store. The proposed approach leveraged knowledge available about the apps on search engines (e.g., Google) and their contextual features (usage patterns), extracted from the device logs of app users. These features were then combined using a Maximum Entropy model for training a classifier. A dataset of 680 apps containing device logs of 443 users was used to evaluate the proposed approach. The results showed that the proposed classifier outperformed other approaches based on topic modeling and word vector analysis.

Berardi et al. [31] proposed a technique for classifying mobile apps into 50 customer defined classes. The authors crawled Google Play and the Apple App Store to extract apps meta-data, including their descriptions, categories, names, ratings, and size. A Support Vector Machines (SVMs) classifier was then trained based on the extracted features. All the selected features were weighted by using the BM25 function [41]. Evaluating the proposed approach over a dataset of 5,792 apps resulted in an F_1 score of 89%.

Sanz et al. [36] proposed an approach for categorizing apps in the Android app store. The objective was to organize the Android market and detect malicious apps. The proposed approach utilized features extracted from the source code of apps, their

requested permissions, and meta-data, including their ratings, size, and advertised permissions. Multiple classification algorithms were then used to classify a dataset of 820 apps sampled from 7 different categories of Google Play. The results showed that Bayesian networks outperformed other algorithms with an Area Under the Curve (AUC) of 93%.

Lulu and Kuflik [32] used unsupervised machine learning to cluster apps based on their functionalities. Specifically, app features were extracted from their app store descriptions and then enriched by content from professional blogs. App features were then represented using Term Frequency - Inverse Document Frequency (TF.IDF) weighted vectors of words. Synonymy relations were resolved using WordNet, a lexical database of semantic relations between words. The authors then used hierarchical clustering to generate hierarchies of functionally-related apps. The effectiveness of the proposed approach was demonstrated on a dataset of 120 apps sampled from Google Play.

Mokarizadeh et al. [33] employed Latent Dirichlet Allocation (LDA) to categorize Android mobile apps. Specifically, LDA was used to model app descriptions (features) [42] and K-means was then used to group similar apps together based on their topic models. The proposed approach was applied to two datasets of Android apps. The results revealed that the default categorization in Google Play did not group apps with similar topics together. Similar to this work, Vakulenko et al. [27] also used topic modeling to group similar apps in the Apple App Store. The authors used LDA to identify recurrent topics in app descriptions. Apps were then classified based on their topic models into 66 categories adapted from the categories and subcategories of the Apple App Store. The results showed that extracted topics extended the original App Store categories and provided in-depth insights into the content of different categories.

Nayebi et al. [43] also leveraged LDA to extract topics from the descriptions of mobile apps. The authors further considered the number of downloads, the number of reviews, and the average ratings as app classification features. DBSCAN was then used to cluster apps into different categories. The proposed technique was evaluated using a dataset of 940 open source apps, sampled from F-Droid. The results showed that DBSCAN performed better in producing homogeneous clusters of apps when using the market attributes of apps (e.g., ratings, downloads, and file size) rather than the topics extracted from their descriptions.

Al-Subaihin et al. [34] proposed a novel approach for app clustering based on their textual features. App features were extracted from their app store descriptions using information retrieval augmented with ontological analysis. Specifically, NLTK's N-gram Collocation Finder was used to extract lists of bi- or tri-grams of commonly collocating words, or *featurelets*, such as <*view*, *image*> or <*send*, *message*>. Agglomerative Hierarchical Clustering (AHC) was then used to cluster apps based on their extracted featurelets. The similarity of feature words was estimated using WordNet. The proposed approach was evaluated using 17,877 apps mined from the BlackBerry app store and Google Play. The cohesiveness of generated clusters was then assessed by human judges. The results showed that the proposed technique improved the categorizations available in modern app stores.

In a more recent work, Al-Subaihin et al. [44] conducted an empirical comparison of text-based app clustering techniques, including topic modeling (LDA) and keyword feature extraction methods [45]. The analysis was conducted using a dataset of 12,664 mobile app descriptions extracted from Google Play. The results showed that, in terms of quantitative cluster quality, LDA-based solutions performed the best.

Gorla et al. [46] proposed CHABADA, a technique for identifying inconsistencies between the advertised behavior of Android apps and their implemented behavior. The authors leveraged LDA to extract topics from the descriptions of mobile apps. The extracted topics were fed into a K-means algorithm to form distinct clusters of apps. Within each cluster, sensitive APIs governed by user permissions were identified. Outliers with respect to API usage were detected using SVMs. These outliers were considered potentially malicious activities. CHABADA was tested on a dataset of 22,500+ Android apps. The prototype was able to detect several anomalies and flag 56% of novel malware.

2.2.2. Motivation

The problem of app classification will persist as long as the number of apps in app stores continues to grow. The search engines of popular app stores used to provide adequate accessibility to apps [47]. However, after the explosive growth in the mobile app market in recent years as well as the constant changes in app store ranking policies, relying on a general keyword search can no longer guarantee equal access to apps [48, 49]. This can have catastrophic impacts on the discoverability of apps, and thus, their survivability. Dynamic app classification engines can mitigate this problem by providing a basis for building new independent app search frameworks that can help different user populations (e.g., health professionals, educators, and businessmen) to find apps that fit their specific needs. This can be particularly important in domains such as Health&Fitness, where recent evidence has shown that having access to the right app can help the quality of health among the general public and help health professionals to communicate better with their patients [50, 51].

Rigorous classification techniques can also provide researchers, across a broad range of disciplines (e.g., business, education, and gaming, etc.), with a framework to automatically zoom-in into their specific domains of interest and get unique in-depth insights into the evolution of apps in such domains in terms of features and user goals. Furthermore, app developers can use such techniques early in the process to explore their ecosystem of competition, or any apps that share their specific set of features. Understanding the domain of competition is crucial for app success and survival [52].

2.2.3. Limitations of Existing Solutions

Our review of related work has exposed several limitations affecting existing app classification solutions. These limitations can be described as follows:

- **Classification features:** A plethora of classification features are used to classify apps. These features are often extracted from the textual descriptions of apps [32, 33, 34, 44, 43], their available meta-data (e.g., ratings, price, etc.) [31, 36], their source code [36], the APIs they use [46], and in some cases, their usage data [35]. In general, going beyond publicly available data can generate unnecessary complexities. For instance, meta-data of apps provide little to no information about their features, and their source code is not always available, and sometimes, obfuscated. In addition, collecting app usage information can raise major privacy concerns, especially if such data is being collected at a large scale.

- **Classification models:** Existing research showed that, due to vocabulary mismatch problems, relying solely on the syntactic attributes (words) of app descrip-

tions, using techniques such as VSM, can generate suboptimal models [44]. Therefore, semantically enabled techniques, such as topic modeling, are commonly used to generate semantic representations of app descriptions. However, such techniques suffer from high operational complexity. For instance, the topic modeling technique LDA requires tuning multiple hyper-parameters in order to generate cohesive topics. These parameters are determined based on heuristics or using the default values provided by tools such as Gensim [53]. Furthermore, such techniques often suffer when dealing with smaller text snippets such as apps descriptions. Due to these limitations, in most cases, the probabilistic distributions of generated topics are not reflective of actual feature topics. Similar problems can be detected in clustering techniques, such as AHC and DBSCAN, where the number of clusters has to be optimized based on subjective measures of cohesiveness.

- **Classification labels:** In the majority of existing work, alternative *ad-hoc* categorizations, or classification labels, are proposed by researchers to be used as *ground-truth* to assess the performance of classification algorithms [34, 35]. However, such labels are often subjective, and in many cases ignore aspects of apps that domain experts, such as doctors, educators, and gamers, might find important for their target user population.

To overcome these limitations, in this chapter we propose a new approach for classifying mobile apps. Our approach uses word embeddings as an underlying technique to generate semantic representations of app descriptions. For our classification categories, or ground-truth, we utilize expert-generated classifications of apps. These classifications are

independently produced by experts with the intention of making apps more accessible to their target users in their domains of operation.

2.3. Word Embeddings

Word embeddings are a type of semantically-aware word representation that allows words with similar meanings to have similar representations. This unique interpretation of text builds upon the distributional hypothesis of Harris, which states that semantically-related words should occur in similar contexts [54]. Formally, a word embedding is a word vectorization technique which represents individual words in a corpus using multi-dimensional vectors of numeric values that are derived from the intrinsic statistical properties of the corpus. Words that have similar meanings should have similar vectors (closer in the vector space). These dense representations proved to be effective for calculating similarities between words using vector geometry, allowing basic computations on these words (low-dimensional matrices) to yield meaningful results (e.g., *India* - *Delhi* $\approx$ *France* - *Paris*, both of these vector subtractions encode the concept of *Capital*) and facilitating more effective automated solutions (e.g., deep learning) for challenging natural language processing (NLP) problems, including document classification [55], sentiment analysis [56], and text summarization [57]. Word2Vec [58], GloVe [38], and fastText [59] are among the most commonly used models of word embeddings [60]. In what follows, we describe these models in greater detail.

2.3.1. Word2Vec

Introduced by Mikolov et al. [58], Word2Vec is a two-layer neural network that utilizes one of two models to produce word embeddings, a Continuous Bag-Of-Words model

(CBOW) and a Skip-gram model. The CBOW model, depicted in Fig. 2.1-a, predicts a word given its surrounding context, while the Skip-gram model, shown in Fig. 2.1-b, uses a word's information to predict its surrounding context. The context of a word w_i is defined by its neighbor words, composed of k words to the left of w_i and k words to its right. k is a hyperparameter of the model, known as the *window size*. Word2Vec predicts the probability that the word w_i is in the context of w_j with the following softmax equation:

$$p(w_i|w_j) = \frac{exp({V'_{wi}}^T V_{wj})}{\sum_{l=1}^{V} exp({V'_{wl}}^T V_{wj})} \tag{2.1}$$

where V'_{wi} is the output vector representation of the target word w_i, V_{wj} is the input vector representation of the word w_j, and V is the vocabulary size. The exp and T stand for exponential and transpose, receptively. The most commonly used pre-trained Word2Vec is learned on more than 100 billion words from the Google News dataset. This model includes 300-dimensional vectors of 3 million words and phrases.

2.3.2. Global Vectors (GloVe)

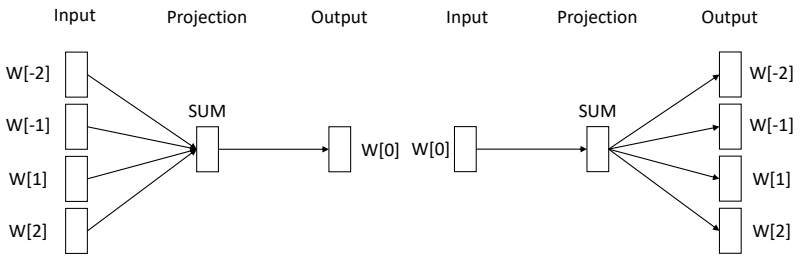
Introduced by Pennington et al. [38], GloVe uses the similarities between words as an invariant to generate their vector representations, assuming that words that occur in similar contexts are more likely to have similar meanings. Similar to Word2Vec, GloVe generates a numeric vector representation of words to preserve their contextual information. However, unlike the predictive-based model of Word2Vec, GloVe is a count-based model. Specifically, GloVe initially constructs a high dimensional matrix of words co-occurrence. Dimensionality reduction is then applied to the co-occurrence count matrix of the corpus. In this matrix, each row shows how often a word co-occurs with other words in a predefined context window in a large corpus. By applying a matrix factoriza-

tion method on the count matrix, a lower dimension matrix is produced, where each row is the vector representation of a word. The dimension reduction approach aims to minimize the *"reconstruction loss"*, thus yield the best lower-dimension matrix that can explain most of the variances in the original matrix and capture the statistics of the entire corpus in its model.

The most commonly used pre-trained GloVe is trained over a six billion token corpus. This corpus was constructed using a combination of *Wikipedia 2014*, which had 1.6 billion tokens, and *Gigaword 5*, which had 4.3 billion tokens. The context window size is set to 10. The vocabulary dictionary of this dataset contains 400,000 most frequent words. This pre-trained model represents word vectors in four dimensions: 50, 100, 200, and 300. Several studies showed that GloVe outperformed Word2Vec and other dimensionality reduction baselines, such as Singular Value Decomposition, over many tasks, including estimating word similarity and Named Entity Recognition [60].

2.3.3. fastText

fastText [59] is another word-embedding model that was developed by Facebook AI in 2016. This model is based on Word2Vec Skip-Gram model. One of the main advantages of fastText is the fact that it considers the internal structures of words to generate their embeddings. In particular, unlike Word2Vec, which takes individual words as input, fastText breaks words into character n-grams. The vector representation of a single word is generated by averaging the vectors of its n-grams. For instance, the word vector of *"diet"* is a sum of the numerical representations of the n-grams: *"di"*, *"die"*, *"diet"*, *"ie"*, *"iet"*, and *"et"*. Using these n-grams, fastText can generate embeddings for words that do not



(a) The CBOW Model (b) The Skip-Gram Model

Figure 2.1: The two architectures of Word2Vec [58].

exist in the original corpus. Multiple pre-trained models of fastText are available. In our analysis, we used the Wiki-news vector representation model which generates one million word vectors learned on *Wikipedia 2017*, *UMBC corpus*, and *statmt.org*. This model contains 16 billion tokens, each represented as a numerical vector of dimension 300.

2.4. Data and Oracle

In the context of supervised data mining, the term oracle refers to *"any mechanism, manual or automated, for determining the ground truth associated with inputs to be classified"* [61]. In this section, we describe our research oracle, including our data collection process, expert-categorizations, and ground-truth generation.

2.4.1. App Data Collection

To collect our data, we developed a Python crawler to automatically scrape app descriptions from the Apple App Store. The dataset used in our analysis was collected in July of 2019. To extract app information, we crawled the web interface of iTunes. The Apple App Store lists all apps classified under each category in alphabetical order. The HTML pages associated with each letter were scraped to extract listed app's URL and iTunes ID. In total, 1,893,256 apps' IDs were scraped. This process is shown in Fig. 2.2.

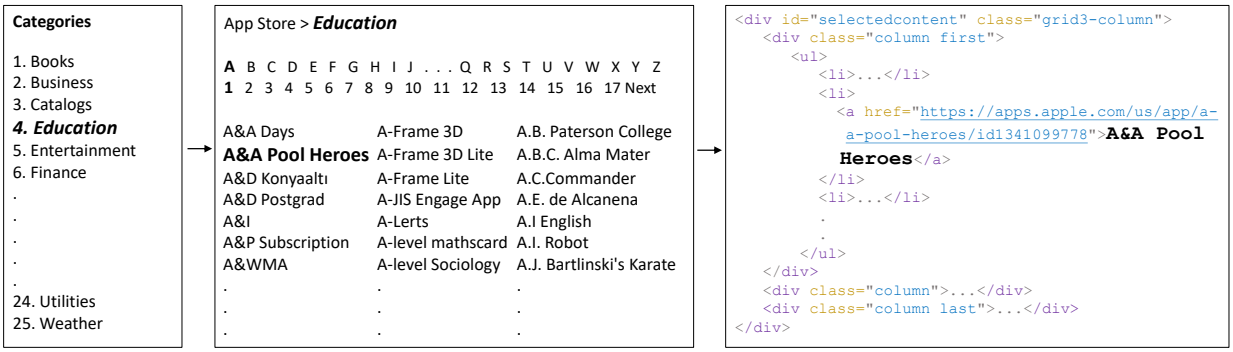


Figure 2.2: Scraping individual app IDs from the HTML pages of each app category.

In the next step, we developed another Python crawler to extract apps' meta-data, including name, description, category, price, and rating average. For each of the scraped app IDs, the crawler requested each app's web page. The crawler then extracted each app's meta-data by parsing the HTML data of its web page. We used a language detection library to detect the app's description language and exclude non-English apps. In total, the meta-data of 1,479,203 English apps were collected. Fig. 2.3 shows this process.

2.4.2. Expert Categorization

One of the main limitations of existing work on app classification is the lack of an expert-verified oracle (e.g., alternative categorization) of the apps being classified. In general, oracles in existing research are either generated by researchers [31, 34] or based on the default categorizations of app stores [62, 36]. To overcome these limitations, in our analysis, we used two existing expert-generated categorizations of apps in the Education and Health categories. The first categorization was introduced by Cherner et al. [28]. In their work, Cherner et al. utilized qualitative research methods to classify Education apps into several categories and subcategories based on their purpose, content, and value. These categories can be described as follows:

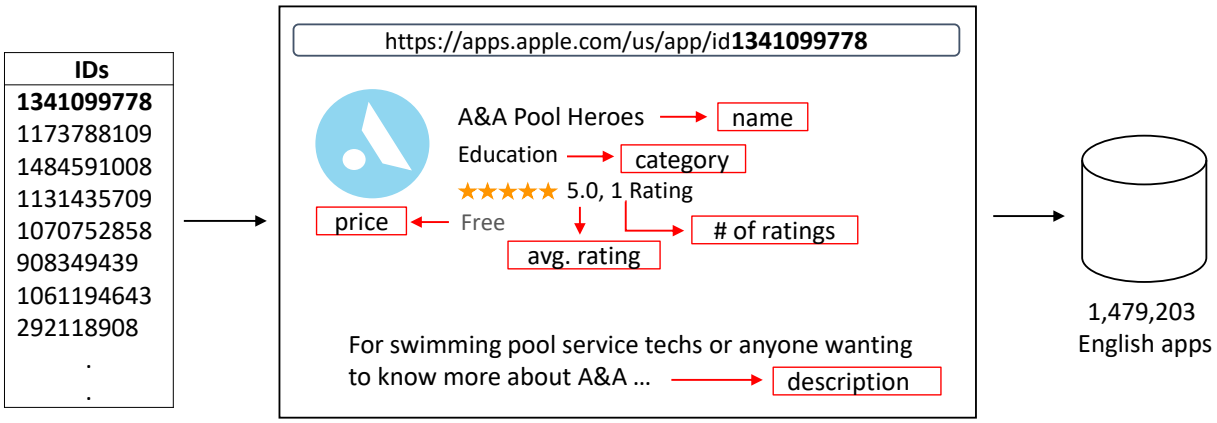


Figure 2.3: Scraping app meta-data from each app's HTML page.

- **Skill-based**: This category includes educational apps that use rote memorization to help students build specific skills, such as literacy, numeracy, science, subject area, reading, and test preparation. Examples of popular apps under this category include, *Vocabulary Builder*, *Khan Academy*, and *LearnEnglish Grammar*.

- **Content-based**: Apps under this category provide access to educational data. These apps are further divided into two groups: subject_area and reference. Subject_area apps contain static pre-programmed educational content. Reference apps, on the other hand, allow users to explore a variety of topics. Examples of popular apps under this category include, *Wikipedia*, *Google Earth*, and *Dictionary*.

- **Function-based**: Apps under this category help transforming the learned content into usable formats. These apps are used for note-taking, presentation, organizing graphics, following school news, and many other functional learning activities. Examples of popular apps under this category include, *Edmodo*, *Inkflow*, and *Remind: School Communication*.

- **Games**: The category of games includes any app that provides some sort of educational content in the form of a game. The Apple App Store labels these apps under both Games and Education categories. Puzzles, trivia, and brain training games are examples of such games. Examples of popular apps under this category include, *Toddler puzzle games for kids*, *MentalUP*, and *Math Ninja*.

- **Misfits**: This category includes apps that do not fit in any of the above categories. These apps are listed under more than one category of the Apple App Store and often have limited educational merit. Examples of popular apps under this category include, *Charades*, *IQ Test: The Intelligence Quiz*, and *Cat sounds effects*.

For our second oracle, we used the framework proposed by Yasini and Marchand [29] to classify health-related apps. In their framework, the use-cases of apps were extracted by a team of IT professionals and medical doctors. The authors then introduced 31 different use-cases which were then grouped into six major usage categories. These categories can be described as follows:

- **Consulting medical information references:** This category includes apps that provide guidelines, scientific popularization, health news, medical textbooks, and access to medical databases. Examples of apps under this category include, *Human Anatomy Atlas, Headspace,* and *Tasty*.

- **Educational tools:** This category includes apps that provide sample educational questions, serious gaming, and access to clinical cases. Examples of popular apps under this category include, *TEAS Mastery, Curiscope,* and *Fig. 1 - Medical Cases*.

- **Health related management:** These apps are used for locating health services, managing drug stocks, and interacting with health-related institutions (e.g., scheduling an appointment, ordering a drug, or connecting to an insurance account). Examples of popular apps under this category include, *GoodRx,* *FollowMyHealth,* and *myHP*.

- **Fulfilling a contextual need:** This category includes apps that collect and interpret medical and health data, check patient records, help diagnosing illnesses, and provide health-decision support. Examples of popular apps under this category include, *iThermonitor, MyFitnessPal,* and *Nike Run Club*.

- **Communicating and/or sharing information:** These apps provide communications platforms for patients, health professionals, and institutions. Examples of apps under this category include, *GAIN Trainer, Coach's Eye,* and *TigerText*.

- **Managing professional activities:** These apps help health professionals to calculate expenses and fees, manage their schedules, and search for jobs. Examples of popular apps under this category include, *Kareo, Medscape,* and *Amion*.

2.4.3. Ground-Truth

Our crawled dataset of apps data contains 138,095 apps from the Education category and a total of 81,944 health-related apps from the Health&Fitness and Medical categories of the Apple App Store. To create our ground-truth, we randomly sampled 300 educational apps and 300 health-related apps. Using the entire content of the app store as our population helped to mitigate the popular app sampling problem [63]. The sample of

300 apps for both datasets is representative at a 95% confidence level with 5.6512% confidence interval. The descriptions of our sampled apps were then manually examined by three judges and then classified into the different categories identified in our oracle. This process can be described as follows:

- An initial meeting was held to discuss the task of the judges.
- Three judges, including two Ph.D. students and a Master's student in software engineering independently classified the apps.
- The manual classification process was carried out over three 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 [64].
- A majority voting was used to determine the final app categories.
- Conflicts ($\sim$5%) were resolved by referring to the original description of the different expert-generated categories as well as the app descriptions. In some cases, apps were installed to get a better sense of their actual functionalities.
- The ground-truth was verified by a fourth judge, a software engineering professor.

The classification process took place prior to conducting the research. Two of the first three judges were not aware of the purpose of the study. On average, the judges had an average of four years of experience in mobile app design and development. Tables 2.1 and Table 2.2 show the number of apps classified under each category and subcategory in each of our domains.

Table 2.1: The number of apps classified under each category and subcategory of educational apps.

Category	**Subcategories**	**Total**
Skill-based	Literacy (38), Numeracy (22), Test-preparation (12), Subject-area (11)	83
Content-based	Subject-area (48), Reference (31)	79
Function-based	Learning-community (58), Others (8)	66
Games	Subject-area (19), Puzzle (14), Brain-training (12)	45
Misfit		27

Table 2.2: The number of apps classified under each category of health-related apps.

Category	**Total**
Consulting medical information references	72
Educational tools	23
Health-related management	80
Fulfilling a contextual need	93
Communicating and/or sharing information	26
Managing professional activities	6

2.5. Approach and Analysis

Our proposed approach (Fig. 2.4) can be broken down into three main steps: data pre-processing, vectorization, and classification. In what follows, we describe each of these steps in greater detail.

2.5.1. Pre-processing

Combinations of text pre-processing strategies are often used in text classification tasks to remove potential noise and to enhance the prediction capabilities of the classifier [65]. In our analysis, app descriptions were first converted into lower case tokens. Tokens that contained non-ASCII characters, digits, and URLs, were removed. English stop-words (e.g., *the, in, will*) were also removed based on the list of stop-words provided in NLTK [66]. The remaining words were then lemmatized. We selected lemmatization over

stemming to preserve the naturalness of words. In particular, stemmers (e.g., Porter stemmer [67]) tend to be prone to over-stemming which happens when too much of the word is removed that the outcome is not a valid natural word (e.g., *general* and *generous* are stemmed to *gener*). This can be a key factor in the performance of methods that use English corpora for similarity calculations.

2.5.2. Vectorization

Under this step, we converted the list of pre-processed tokens in each app's description into a vector of word embeddings using the pre-trained models of Word2Vec, GloVe, and fastText. We then used the generated embeddings to represent the whole description. Word collection (e.g., phrase, sentence, or paragraph) embeddings can be computed using different methods, such as unweighted averaging [68], Smooth Inverse Frequency (SIF) [69], Doc2Vec [70, 71] and Recursive Neural Networks (RNNs) [72]. In our analysis, we used the simple unweighted averaging method to obtain an embedding for each app description. Averaging word vectors has been proven to be a strong baseline for paragraph representation especially in cases when the order of words is unimportant [69, 73, 74]. Formally, the vector representation (V_D) of the app D, can be computed as:

$$V_D = \frac{1}{n} \sum_{i=1}^{n} V_{w_i} \tag{2.2}$$

where the description is composed of words $w_0, w_1, ..., w_n$. Each word is represented as a vector $V_{w_0}, V_{w_1}, ..., V_{w_n}$ of word embeddings. Table 2.3 shows the different pre-processing steps and the vectorization step (using Word2Vec, GloVe, and fastText) being applied to the sample app description sentence, *" dictionary® series is designed to make it easier to learn and use the technical jargon, and abbreviations."*

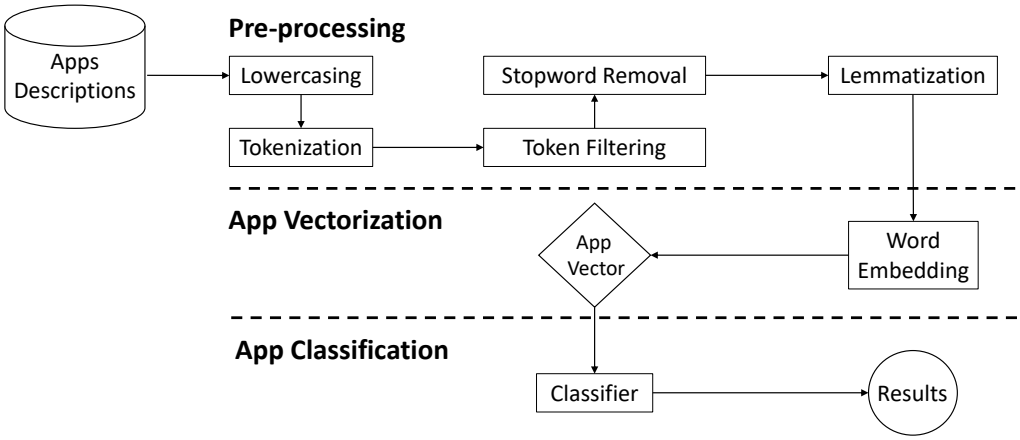


Figure 2.4: The main steps of the proposed approach.

2.5.3. Experimental Baselines

In addition to our word embedding vectors, we generate three other types of vectors for app descriptions. These vectors will be used as experimental baselines to compare the performance of word embeddings. These representations can be described as follows:

- **Vector Space Model:** VSM is an algebraic model that consists of a single term-document matrix. Each row of the matrix represents a single term found in the corpus and each column represents an individual document. Each entry in the matrix $w_{i,j}$ is the weight of the term j in the document i, indicating the importance of the term to the document's subject matter. While the raw frequency of the term in the document can be used as a weight, another approach, known as term frequency-inverse document frequency (TF.IDF) is typically used [75]. TF.IDF is calculated as the product of the frequency of the term in the document (TF) and the term's scarcity across all the documents (IDF). Formally, TF.IDF can be computed as:

$$TF.IDF = f(w,d) \times \log \frac{|D|}{df(w)} \tag{2.3}$$

where $f(w, d)$ is the term frequency of the word w in document d, D is the total number of documents in the corpus, and $df(w)$ is the number of documents in the corpus D that contain the word w. In this chapter, we vectorize each app's description as the TF.IDF of its words. In particular, each app is represented as a vector of size $|V|$, which is the number of words in the description (i.e. $V = \{w_0, w_1, ..., w_m\}$). The i^{th} entry of this vector is set to $TF.IDF(w_i)$ if the description contains the word w_i, and 0 otherwise.

- **Latent Dirichlet Allocation:** LDA is an unsupervised probabilistic approach for estimating a topic distribution over a corpus [42]. A topic consists of words that collectively represents a potential thematic concept [42, 76]. 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 distribution 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 [77]. 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 [42].

We use Gensim, a Python-based open-source toolkit for vector space modeling and topic modeling, to extract topics from our dataset of apps descriptions [53]. LDA's hyper-parameters α and β are calibrated based on the heuristics that are commonly used to calibrate topic modeling in text analysis [78, 79]. In particular, α is set to be automatically learned from the corpus and β is set to 1/(number of topics). The number of iterations for the sampling process is set to 1000 to ensure the stability of generated topics [77]. The number of topics to be found by LDA, k, is set to the number of classification labels. When applied to descriptions, LDA represents each description as a vector of size k. The i^{th} entry of this vector is set to the probability of the topic i to be present in the description.

- **BM25:** BM25 is a text scoring method that was introduced in 1994 as a robust variant of the TF.IDF method [41]. BM25 is calculated as the product of modifications of TF and IDF. Formally, BM25 can be computed as:

$$BM25(w_i) = \frac{f(w,d) \times (k+1)}{f(w,d) + k \times (1 - b + b \times \frac{|d|}{avgD})} \times log(1 + \frac{|D| - df(w) + 0.5}{df(w) + 0.5}) \quad (2.4)$$

where $f(w,d)$ is the term frequency of the word w in document d, D is the total number of documents, and $df(w)$ is the number of documents that contain the word w, $avgD$ is the average length of the documents, b is a parameter to tune the impact of the document length, and k is a parameter to tune the impact of term frequency. BM25 has been used in the literature to vectorize app descriptions [31]. Each description is represented as a vector of size $|V|$, where $|V|$ is the number of words in the description (i.e. $V = \{w_0, w_1, ..., w_m\}$). The i^{th} entry of this vector is set to $BM25(w_i)$ if the description contains the word w_i and 0 otherwise.

Table 2.3: An example of the text pre-processing steps used in our analysis.

Sentence	*"dictionary® is designed to make it easier to learn, use the technical jargon, and abbreviations"*	
Good tokens	[*dictionary, is, designed, to, make, it, easier, to, learn, use, the, technical, jargon, and, abbreviations"*]	
SW removal	[*dictionary, designed, make, easier, learn, use, technical, jargon, abbreviations*]	
Stemmer	[*dictionari, design, make, easi, learn, use, technic, jargon, abbrevi*]	
Lemmatizer	[*dictionary, design, make, easy, learn, use, technical, jargon, abbreviation*]	
Word embedding	dictionary	[-0.84041, -0.11159, 0.49872, 0.30307, . . . , -0.08523, 0.80811, -0.12826, 0.088422]
	design	[0.28934, 0.026391, -0.5053, -0.9966, . . ., 0.44844, 0.063997, 0.56751, 0.059809]
	make	[.3547e-01, 1.1550e-01, -2.9983e-01, . . ., -3.9427e-01, -2.3503e-01, 3.0761e-01]
	easy	[6.7032e-02, -1.0813e-01, 4.4981e-01, . . ., -1.5073e-01, -2.5662e-01, 8.0550e-02]
	learn	[-3.3157e-01, -8.8796e-02, -1.6376e-01, . . ., 1.2686e-02, -8.1500e-02, 1.3113e-03]
	use	[-1.0515e-01, 1.3407e-01, 1.3839e-01, . . ., -4.1949e-01, 8.9402e-02, 1.7569e-01]
	technical	[0.32273 , -0.085892, -0.26033, -0.3997, . . ., -0.78574, -0.23319, -0.11036, -0.59291]
	jargon	[-0.54072, -0.27198, 0.34199, 0.17347, . . ., -0.46058, 0.30216, -0.266, 0.19508]
	abbreviation	[-0.1607 , 0.15862 , 0.77783 , -0.16962 , . . ., -0.12764 , -0.75467 , 0.34786 , -0.31688]
Embedding		[-1.12256169e-01, -4.28820204e-04, 1.39360920e-01, . . ., 8.19473062e-03, -1.63306966e-02]

2.5.4. Classification and Evaluation

Under this step, we classify apps in our dataset using the different vectorization techniques presented earlier. Our classification configurations can be described as follows:

- **Classification algorithms**: To classify our data, we experiment with multiple classification algorithms: Naive Bayes (NB) [80], Support Vector Machines (SVM) [65], Random Forests (RF) [81], Decision Trees (DT) [82], AdaBoost [83], and Logistic Regression [84]. These algorithms are commonly used to classify crowd feedback in the mobile app market [85, 86, 87]. Their success can be attributed to their ability to deal with short texts (e.g., tweets, user reviews, YouTube comments, etc.) [88, 89]. Our analysis is performed using Scikit-learn, a Python library which integrates a wide range of state-of-the-art machine learning algorithms for supervised and unsupervised classification problems [90].

  We use a one-vs-one strategy for our multi-class classification. This classification strategy splits a multi-class classification into one binary classification problem per

pair of classes. The class that receives the majority of votes is selected as the predicted class. Hyperparameter tuning is used to ensure that each classifier achieves its best possible prediction given the data [91]. Our specific list of hyperparameters is shown in Table 2.4. We use RandomizedSearchCV, a methodology which uses cross-validation to optimize the hyperparameters of the classifier. A detailed explanation of each parameter can be found on Scikit-learn's webpage [92].

- **Classification features**: We extract app classification features from their descriptions. Each app's description is vectorized using the vectorization techniques described in Section 2.5.2. For each app description, word embedding methods generate a d-dimensional feature vector. The size of the vector for Word2Vec and fastText is set by default to ($d = 300$). GloVe, can generate different size vectors, where ($d \in \{50, 100, 200, 300\}$), VSM generates vectors of size $|V|$, where $|V|$ is the number of words in the description (i.e. $V = \{w_0, w_1, ..., w_m\}$). Using LDA, each app is vectorized into a feature vector of size k representing the probabilistic distribution of the description over the set of k LDA topics. We further analyze the impact of adding existing meta-data features (i.e. the number of ratings, average rating, app size, category, and price) on the classification accuracy. Therefore, for each vectorization technique, we append the extracted meta-data to the vectorized representation of each app.

- **Training settings**: To train and test our classifiers, we use *10*-fold cross-validation. This approach creates 10 partitions of the dataset. In each partition, 90% of the instances are considered as the training set and 10% as the test set. 10-

fold cross-validation is selected over other techniques, such as the holdout method (e.g. train/test split), to decrease the variance of the results.

- **Validation metrics**: The standard measures of precision (P), recall (R), and F-measure (F_β) are used to evaluate the performance of our classification algorithms. These measures are computed independently for each classification label and averaged over all the labels. Precision is calculated as the ratio of the number of correctly classified instances under a specific label (t_p) to the total number of classified instances under the same label ($t_p + f_p$). Recall is calculated as the ratio of t_p to the total number of instances belonging to that label ($t_p + f_n$). The F-measure represents the weighted harmonic mean of precision and recall. β is used to emphasize precision or recall. A $\beta = 2$ is commonly used in related literature to slightly emphasize recall over precision. Formally, f_β can be calculated as $(\beta^2 + 1)PR/(\beta^2 P + R)$.

2.5.5. Results and Analysis

The results of classifying our sets of Education and Health apps are shown in Table 2.5. On average, our classification algorithms achieved their best performance when app descriptions were vectorized using GloVe_{300}. In particular, SVM (linear kernel) was able to achieve the best results in separating the general categories of education apps, achieving an F_2 of 0.84. Logistic Regression achieved a comparable performance ($F_2 = 0.8$) . However, the accuracy went down when we classified education apps at a subcategory level. This was actually expected given that it becomes harder to separate categories at such a granularity level.

Table 2.4: Hyperparameter configuration for each classifier.

Classifier	**Parameter set**
Naive Bayes	Gaussian NB: *'var_smoothing'* $\in \{10^{-10}, 10^{1}\}$
	Multinomial NB: *'alpha'* $\in$ {0, 1}
Adaboost	*'n_estimators'* $\in$ {10, 50, 100, 200}
Random Forest	*'max_depth'* $\in$ {10, 30, 50, 100, None}
	'max_features' $\in$ {'auto', 'sqrt'}
	'min_samples_split' $\in$ {2, 5, 10}
	'n_estimators' $\in$ {10, 50, 100, 200}
KNN	*'k'* $\in$ {2, 5, 7, 10}
SVM	*'kernel'* $\in$ {'linear', 'rbf'}
	'C' $\in$ {0.1, 1, 10, 100}
	'gamma' $\in$ {1/(n_features * X.var()), 1/n_features}
Decision Trees	*'max_depth'* $\in$ {10, 30, 50, 100, None}
	'max_features' $\in$ {'auto', 'sqrt'}
	'min_samples_split' $\in$ {2, 5, 10}
	'criterion' $\in$ {'qini', 'entropy'}
Logistic Regression	*'penalty '* $\in$ {'l1', 'l2'}
	'C' $\in$ {0.1, 1, 10, 100}

In the health dataset, SVM was also able to achieve the best results ($F_2 = 0.8$). Logistic Regression ($F_2 = 0.78$) and KNN ($F_2 = 0.78$) were able to achieve comparable performance. However, the accuracy was on average lower than the accuracy achieved on the education dataset. A comparison of the performance based on the different size vectors generated by GloVe is shown in Fig. 2.5. In general, for both datasets, GloVe achieved its best results at vector size 300. Increasing the size of vectors resulted in more expressive vectors that capture all word relations.

Our results also showed that adding app meta-data to the set of classification features did not improve the performance. As Fig. 2.6 shows, apps meta-data failed to enhance the predictive capabilities of our classifiers. To get a better sense of our results, we compared our findings with Berardi et al. [31]. In their work, the authors considered app descriptions as well as apps' meta-data (rating, size, category, and price) as classification

Table 2.5: The performance (Precision (P), Recall (R), F-measure (F_2)) of the different classification algorithms using the different app description vectorization techniques.

Approach	**Classifier**	**Education categories**			**Education sub_categories**			**Health apps**		
		P	**R**	F_2	**P**	**R**	F_2	**P**	**R**	F_2
VSM	NB	0.53	0.56	0.55	0.28	0.35	0.33	0.5	0.57	0.55
	AdaBoost	0.37	0.39	0.38	0.28	0.25	0.25	0.33	0.42	0.39
	Random Forest	0.67	0.65	0.65	0.46	0.45	0.45	0.55	0.59	0.58
	KNN	0.65	0.62	0.62	0.56	0.51	0.51	0.64	0.6	0.6
	SVM	0.71	0.69	0.69	0.57	0.5	0.51	0.64	0.66	0.65
	Decision Trees	0.56	0.51	0.51	0.44	0.38	0.39	0.52	0.51	0.51
	Logistic Regression	0.68	0.67	0.67	0.44	0.45	0.44	0.56	0.61	0.59
	Average	0.59	0.58	0.58	0.43	0.41	0.41	0.53	0.56	0.55
LDA	NB	0.27	0.32	0.3	0.15	0.12	0.12	0.41	0.27	0.28
	AdaBoost	0.42	0.35	0.36	0.14	0.2	0.18	0.32	0.29	0.29
	Random Forest	0.49	0.41	0.42	0.29	0.28	0.28	0.33	0.31	0.31
	KNN	0.38	0.35	0.35	0.26	0.25	0.25	0.34	0.28	0.29
	SVM	0.35	0.41	0.39	0.2	0.29	0.26	0.33	0.38	0.36
	Decision Trees	0.44	0.38	0.39	0.29	0.25	0.25	0.3	0.27	0.27
	Logistic Regression	0.35	0.4	0.38	0.23	0.31	0.28	0.33	0.39	0.37
	Average	0.38	0.37	0.37	0.22	0.24	0.23	0.33	0.31	0.31
GloVe 300	NB	0.7	0.68	0.68	**0.67**	**0.63**	**0.63**	0.72	0.66	0.67
	AdaBoost	0.69	0.67	0.67	0.4	0.38	0.38	0.54	0.55	0.54
	Random Forest	0.73	0.71	0.71	0.56	0.53	0.53	0.7	0.71	0.7
	KNN	0.74	0.72	0.72	0.66	0.58	0.59	0.81	0.78	0.78
	SVM	**0.85**	**0.84**	**0.84**	0.6	0.57	0.57	**0.83**	**0.8**	**0.8**
	Decision Trees	0.63	0.59	0.59	0.46	0.44	0.44	0.6	0.58	0.58
	Logistic Regression	0.82	0.8	0.8	0.57	0.54	0.54	0.79	0.78	0.78
	Average	0.73	0.71	0.71	0.56	0.52	0.53	0.71	0.69	0.69
Word2Vec	NB	0.74	0.69	0.69	0.62	0.52	0.53	0.68	0.63	0.63
	AdaBoost	0.55	0.51	0.51	0.21	0.22	0.21	0.4	0.37	0.37
	Random Forest	0.67	0.68	0.67	0.52	0.48	0.48	0.62	0.65	0.64
	KNN	0.64	0.64	0.64	0.53	0.48	0.48	0.64	0.58	0.59
	SVM	0.67	0.68	0.67	0.43	0.45	0.44	0.59	0.64	0.62
	Decision Trees	0.47	0.45	0.45	0.34	0.28	0.29	0.5	0.48	0.48
	Logistic Regression	0.66	0.66	0.66	0.41	0.45	0.44	0.54	0.62	0.6
	Average	0.62	0.61	0.61	0.43	0.41	0.41	0.56	0.56	0.56
fastText	NB	0.68	0.64	0.64	0.6	0.6	0.6	0.72	0.65	0.66
	AdaBoost	0.63	0.57	0.58	0.16	0.21	0.19	0.52	0.6	0.58
	Random Forest	0.69	0.7	0.69	0.57	0.53	0.53	0.66	0.69	0.68
	KNN	0.71	0.68	0.68	0.62	0.57	0.57	0.83	0.76	0.77
	SVM	0.79	0.78	0.78	0.59	0.53	0.54	0.8	0.79	0.79
	Decision Trees	0.61	0.57	0.57	0.45	0.41	0.41	0.5	0.47	0.47
	Logistic Regression	0.78	0.77	0.77	0.49	0.48	0.48	0.65	0.7	0.68
	Average	0.69	0.67	0.67	0.49	0.47	0.47	0.66	0.66	0.66
BM25	NB	0.48	0.33	0.35	0.4	0.32	0.33	0.54	0.39	0.41
	AdaBoost	0.42	0.37	0.37	0.26	0.25	0.25	0.37	0.37	0.37
	Random Forest	0.58	0.57	0.57	0.45	0.39	0.4	0.49	0.56	0.54
	KNN	0.55	0.46	0.47	0.48	0.39	0.4	0.54	0.45	0.46
	SVM	0.53	0.47	0.48	0.51	0.42	0.43	0.52	0.48	0.48
	Decision Trees	0.51	0.47	0.47	0.38	0.33	0.33	0.42	0.41	0.41
	Logistic Regression	0.58	0.58	0.58	0.55	0.45	0.46	0.54	0.55	0.54
	Average	0.52	0.46	0.47	0.43	0.36	0.37	0.48	0.45	0.46

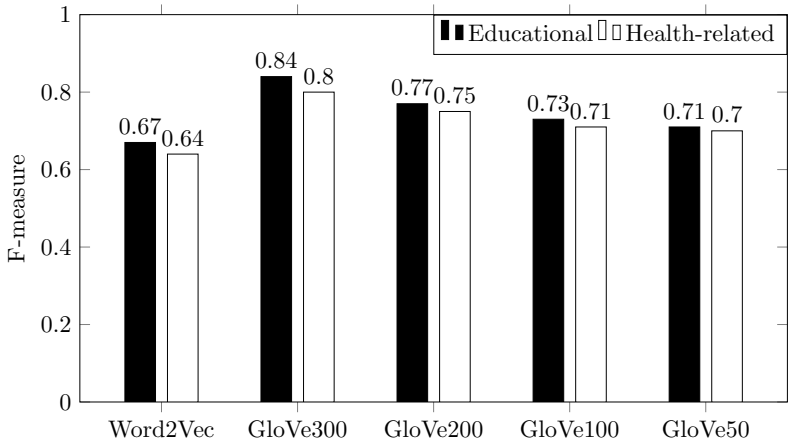


Figure 2.5: SVM classification results using different GloVe size vectors.

features. App descriptions were preprocessed using tokenization, stop-word removal, and stemming and then vectorized using BM25 [93]. We replicated this type of analysis on our dataset. Following Berardi et al. [31], a mutual information-based feature selection method was also applied to select the most informative set of app features [94]. The results showed that adding meta-data as classification features did not improve the results. This was actually expected given that, unlike descriptions, meta-data attributes of apps (apps' names, ratings, downloads) hardly convey any functionality-related information.

2.5.6. Statistical Analysis

We use statistical testing to measure the difference in performance between our proposed approach and other experimental baselines. We first used the Shapiro-Wilk test to test the normality of the variable (f-measure) being compared in our analysis [95]. The results showed that the normality assumption did not hold for the majority of our comparisons. Therefore, for our type of data, we used the non-parametric tests. In particular, we used Wilcoxon signed-rank and Friedman to measure statistical significance of difference between the performance of our proposed approach and other baselines ($p\text{-}value = 0.05$).

To examine the effect of using different vectorization methods (GloVe, Word2Vec, fastText, LDA, TF.IDF, and BM25) on the accuracy of our classifiers, we first applied the Friedman hypothesis test with Bonferroni-Holm correction (control method = GloVe) at *p-value* = 0.05 [96, 97]. Our null hypothesis H_0 states that there is no difference in the f-measures between different vectorization methods. The alternative hypothesis H_1 is in favor of a significant difference between our different methods. To show the effect size of the difference, we used Kendall's W (coefficient of concordance). Kendall's uses the Cohen's interpretation guidelines of 0.1 (small effect), 0.3 (moderate effect), and above 0.5 as a strong effect [98]. The results showed a *p-value* < 0.005, indicating that we can reject H_0 with at least a medium effect size (Kendall's W = 0.47) between the classification results of GloVe and other vectorization techniques. Given these results, we conclude that GloVe leads to statistically significant improvement in comparison to other techniques, including the baseline using app meta-data as classification features.

We further examine the effect of considering apps meta-data as classification features. In particular, we compare the f-measure values of SVMs+GloVe and SVMs+GloVe+metadata features, applying 10-fold cross validation in both cases. Our null hypothesis H_0 states that adding apps metadata does not have any effect on the f-measure. The alternative hypothesis H_1 is in favor of the effect of adding apps metadata. Since in this test we have two groups of data, we use Wilcoxon signed-rank at *p-value* $<$ 0.05 to measure statistical significance. To show the effect size of the difference between applied methods, we calculate Cliff's Delta (d), a non-parametric effect size method. We interpret the effect size values as *small* for $0.147 < d < 0.33$, *medium* for $0.33 < d < 0.474$, and *large* for $d > 0.474$ [99, 100]. Our results show that there is a statistically

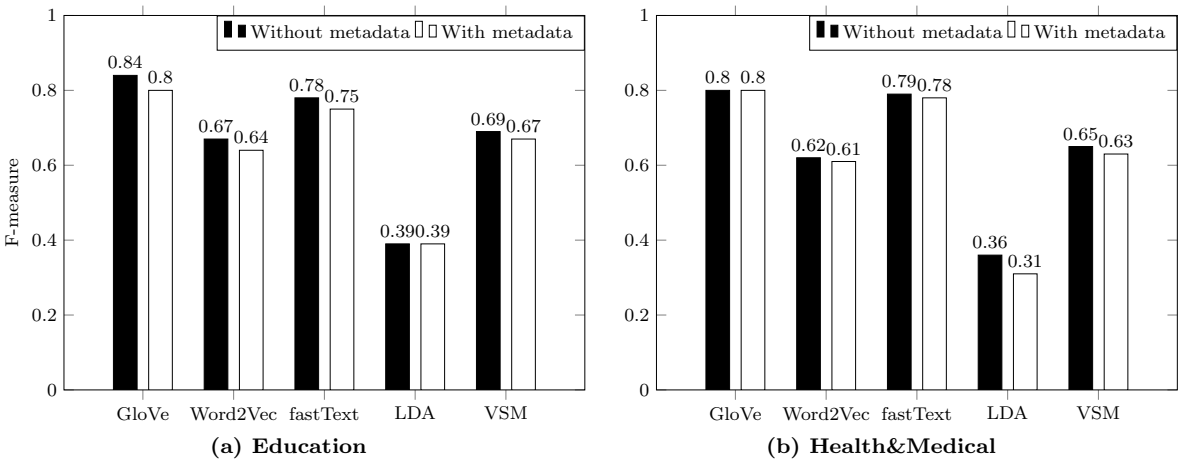


Figure 2.6: The results of classifying educational health-related apps using SVMs with-/without adding apps metadata (i.e. the number of ratings, the average rating, the app size, the category, and the price.

significant difference (p-*value* < 0.05) with at least a medium (d = 0.07) effect size when considering apps metadata as classification features, indicating that adding apps metadata can significantly degrade the accuracy of classification.

2.6. Validation and Human Experiment

In the first phase of our analysis, we showed that word embeddings of app descriptions can classify apps into more focused categories of application domains. To further validate our findings, in this section, we apply our approach to a third dataset of mobile apps, sampled from the domain of Sharing Economy (SE). The Sharing Economy refers to a sustainable form of peer-to-peer business exchange that is built around sharing assets and resources [101]. Over the past decade, SE apps, such as Uber, TaskRabbit, and Airbnb, have caused major disturbances in established classical markets, enabling people to exchange and monetize their underused assets at an unprecedented scale [102, 103, 104]. As of today, there are thousands of Sharing Economy apps, operating in a market sector that is projected to grow to 335 billion U.S. dollars by 2025 [105].

The domain of Sharing Economy presents a prime example of application domains where a more precise categorization of apps is highly needed. In particular, popular app stores, such as the Apple App Store and Google Play, do not provide a separate category for Sharing Economy apps, instead, these apps are scattered over a broad range of categories that hardly capture their core functionalities. For example, both Uber (ride-sharing) and Airbnb (lodging) are categorized under the Travel category in the App Store and Gigwalk, a freelancing app, is classified under the Lifestyle category. This makes it very challenging for service providers and receivers to navigate the landscape of SE apps and make optimized economic decisions in one of the fastest growing software ecosystems in the world.

In this section, we use our classification approach to classify a dataset of popular Sharing Economy apps based on their core functionalities. We then conduct a human experiment, using 12 participants, to validate our classifier from an end-user point of view.

2.6.1. Dataset and Expert Classification

To conduct our analysis, we sample a dataset of SE apps that are currently active in the market. We enforce the following criteria on apps to be included in our sample:

1. The app must facilitate some sort of a P2P connection and include the sharing of some sort of a resource, such as an asset (e.g., an apartment, car, electric drill, etc.) or a skill (e.g., plumbing, hair styling, coding, etc.).

2. The app must be available on Google Play or the Apple App Store so that we can extract its description.

3. The app must be located and/or have a substantial presence in the U.S. By focusing on the U.S. market, we ensure that app descriptions are available in English and that the app supports a service that is familiar to the casual U.S. user.

With these criteria in place, we searched for apps to be included in our dataset. We conducted a Google search using the query: `(sharing OR shared OR gig) AND economy AND (platforms OR apps OR systems)`. We examined the first 10 pages of the search results and added 72 new platforms that matched our inclusion criteria. We then used the *similar* feature on Google Play and the Apple App Store to locate any apps we missed through Google search. Specifically, we examined the list of similar apps resulting from searching app stores for each of our 72 apps. Lightweight snowballing was then used to add any major apps that we might have missed. Apps were iteratively added until no more new apps that satisfied our inclusion criteria were located. In total, 108 unique apps were included in our dataset. Descriptive statistics of our dataset are provided in Table 2.6. This table shows the apps average rating and their number of reviews on Google Play and the Apple App Store.

To generate our expert-based categories, we went through each app in our sample and independently examined their Apple App Store descriptions. We used memoing to keep track of the reasoning behind each suggested category. Axial coding was then used to consolidate individual categories into more abstract categories. For example, the categories of *food delivery* and *grocery delivery* were merged into a single *delivery* category. Generated categories were then iteratively revised until no more categories were found. By the end of our classification process, six main categories of Sharing Economy apps, shown in Fig. 2.7, had emerged. These categories can be described as follows:

- **Skill-based:** These apps facilitate the sharing of personal skills. Specific examples include the baby sitting apps *Sittercity* and *Urbansitter*, the tutoring apps *Verbling*, *Codementor*, and *Classgap*, and the freelancing apps *Fiverr* and *Upwork*.

- **Delivery:** Under this category, we include apps which enable users to utilize their vehicles to deliver goods to other users. Examples of apps in this category include *UberEats*, *Grubhub*, and *Shipt* for grocery and food delivery and *DriveMatch*, *uShip*, and *Dolly* for hiring delivery drivers.

- **Ride-sharing:** This category includes apps which allow their users to share rides, such as carpooling and driver/rider connections. Examples of apps in this category include traditional ride-sharing services, such *Uber*, *Lyft*, and *Via*, as well as more specialized platforms, such as *HopSkipDriver* for children transportation, *Veyo* for medical transportation, and *Wingz* for hiring a driver.

- **Asset-sharing:** Under this category, we include any app which enables users to lend and borrow assets. This category is different from other categories in the sense that the resource being shared is the asset itself (e.g., a vehicle or an electric drill), not the person (e.g., a driver or electrician). Examples of apps under this category include the boat sharing apps *Get-MyBoat* and *Boatsetter*, the bike sharing app *Spinlister*, and the RV sharing apps *RVezy* and *Outdoorsy*.

- **Lodging:** This category contains renting and short-term accommodation services such as *Airbnb*, *Vrbo*, and *Misterbnb* as well as space-sharing for storage (*Neighbor*), events and work (*Splacer* and *LiquidSpace*), and even parking (*ParqEx*).

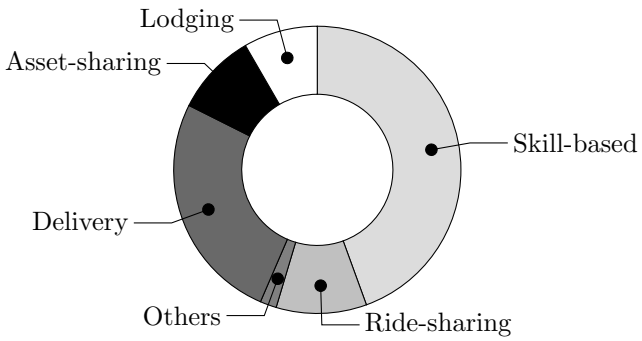


Figure 2.7: Distribution of our apps over application domains.

Table 2.6: Descriptive statistics for the 108 apps in our dataset.

Metric	**Mean**	**Median**	**Min**	**Max**
App Store Rating	4.23	4.60	1.60	4.90
Google Play Rating	3.86	3.90	2.00	4.90
App Store # of Reviews	201K	2.4K	2	8.9M
Google Play # of Reviews	134K	1.3K	7	7.91M
Google Play # of Installs	6.9M	100K	1K	500M

- **Other:** Although our objective was to classify all apps into the main general categories, two apps in our dataset were too niche-oriented to warrant the creation of a separate category. These apps are *Prosper* for lending and borrowing money and *Kickstarter*, a platform for crowdfunding various projects.

2.6.2. Automated Classification

To classify our apps, we extracted their descriptions from the Apple App Store. Descriptions were then processed by applying tokenization, removing non-ASCII characters and stop words, and lemmatization (Sec. 2.5.1). We then vectorized the descriptions using GloVe (vector size = 300). An SVMs classifier (linear kernel) was then trained on the data using 10-fold cross-validation. The results showed that our proposed model (SVMs+GloVe) achieved an F_2 of 0.8 (Precision = 0.84 and Recall = 0.79). Overall, these results came out consistent with our previous results on the education and health datasets.

A closer look at our results revealed that our automated classifier failed to correctly label apps that describe their features using words that are common in more than one category. For example, `ParqEx` is a parking sharing app that was misclassified as a delivery app. `ParqEx` description contains words such as *space*, *parking*, and *book* which are common in both delivery and lodging apps. Another observation is that two of the skill-based apps were misclassified as lodging apps. A possible explanation is that our dataset included only seven lodging apps, which were not enough data for the classifier to generate a separate category for these apps. Our expectation is that such errors will be minimized as more data becomes available for our classifier to work with.

2.6.3. Study Participants and Procedure

To further evaluate the effectiveness of our automated classifier, we conducted a human experiment with 12 study participants. Our participants were recruited through convenience sampling. Our sample has four females and eight males with an average age of 36 (min = 21, max = 48). All participants reported using one or more Sharing Economy apps, either as service providers (e.g., driving for Uber, Lyft, DoorDash and Instacart) or receivers (e.g., renting an Airbnb, ordering groceries through Instacart, and hiring workers through TaskRabbit). Choosing subjects who are average end-users of apps, rather than expert software developers, provides an evidence of the value of our approach to the casual users. Our experimental procedure can be described as follow:

- We randomly sampled 20 apps from our dataset using stratified sampling. We excluded popular apps such as Uber, Lyft, Airbnb, TaskRabbit, and UpWork as classifying these apps can be trivial.

- We prepared an assignment for our study participants. The assignment included a description of our expert-generated categories of Sharing Economy apps as well as the descriptions of the 20 apps in our sample.

- Each of our 12 study participants was presented with the assignment and asked to classify each app by picking a category from the expert-based classification. If they thought that none of our suggested categories was a good fit for the app, they were advised to either classify the app as *others*, or add their own category. Apps were listed in a random order in each assignment. Randomization helped to control any effect that might result from the order of the treatment, such as, our participants getting bored or tired and not spending as much time on classifying apps that appear later in the list.

- The human generated classifications are then compared with our automatically generated classifications and the results are then collected and analyzed.

2.6.4. Results

The results of our study are shown in Table 2.7. The table includes the list of apps used in our study, their ground truth classifications, their classifications by our approach, their current categories in the Apple App Store, and our 12 study participants classification. The results show that, our study participants achieved an average 79% agreement with our automated classification results. Cases of disagreement were detected over apps which were misclassified by our approach. For instance, our study participants had a hard time finding the right category for *Carvertize*, an app which pays drivers by placing adver-

Table 2.7: The results of our human study. We use the following code for the different categories: 1 = Skill-based, 2 = Delivery, 3 = Ride-sharing, 4 = Asset-sharing, 5 = Lodging, and 6 = Other. GT is our ground-truth classification, AT is the automated classification, $\{S_1, \ldots, S_{12}\}$ is the set of subjects.

App	**Category**	**GT**	**AT**	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}
Rvezy	Travel	4	4	4	4	4	4	4	4	4	4	4	4	4	4
Carvertise	Productivity	4	1	2	3	3	4	4	4	3	3	3	3	4	4
SparkDriver	Business	2	2	2	2	2	2	2	2	2	2	2	2	2	2
BiteSquad	Food&Drink	2	2	2	2	2	2	2	2	2	2	2	2	2	2
PointPickup	Utilities	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Shipt	Business	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Pickup	Lifestyle	2	1	2	2	1	1	4	2	2	2	2	2	2	1
ParqEx	Navigation	5	2	5	5	5	5	5	5	4	5	5	4	5	4
Couchsurfing	Travel	5	5	5	5	5	5	5	5	5	4	5	4	5	5
Wingz	Travel	3	3	3	3	3	3	3	1	3	3	3	3	3	3
zTrip	Travel	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Eatwith	Travel	1	5	1	5	5	1	1	1	1	1	1	1	1	1
Withlocals	Travel	1	5	1	1	1	1	1	1	1	1	1	1	1	1
Gigwalk	Lifestyle	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Handy	Lifestyle	1	1	1	1	1	1	1	1	1	1	1	1	1	1
GigSmart	Business	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Bambino	Lifestyle	1	1	1	1	1	1	1	1	1	1	1	1	1	1
UrbanSitter	Lifestyle	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Caregiver	Lifestyle	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Fiverr	Business	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Agreement with automated				0.8	0.85	0.8	0.75	0.75	0.75	0.8	0.75	0.8	0.75	0.8	0.9

tisements on their cars. This app should be classified as an asset-sharing app, however, it was incorrectly labeled as skill-based by our classifier. This can be attributed to the fact that this particular app describes its features using words such as *media*, *student*, *college*, and *specialize*, missing common words in the asset-sharing category, such as *car* and *lend*. This vague terminology confused our classifier as well as most of our participants.

To evaluate the agreement between study participants, we used Cohen's kappa [106], a commonly used measure of the inter-rater agreement [107, 108]. This method is a more robust method than simple percent agreement since it takes into account the agreements occurring by chance [109]. Formally, Cohen's kappa (k) can be calculated as:

$$k = \frac{p_o - p_c}{1 - p_c} \tag{2.5}$$

where p_o is the proportion of items for which the raters agreed on, and p_c is the proportion of items that agreement was expected by chance. Our results of human assessment suggest a Cohen's kappa of **0.83**, which indicates an almost perfect agreement between our study participants. Cohen's kappa values less than zero are interpreted as no agreement, 0.01 – 0.20 as none to slight, 0.21 – 0.40 as fair, 0.41 – 0.60 as moderate, 0.61 – 0.80 as substantial, and 0.81 – 1.00 an as almost perfect agreement between raters [106]. Overall, our human assessment shows that our automated classification approach can generate accurate classifications that correlate with human generated classifications to a large extent.

2.7. Discussion

In this section, we discuss our main analysis results and we provide further analysis on some of our findings in this chapter.

2.7.1. Word Embeddings vs. Topic Modeling

We begin our discussion by comparing the performance of our approach with LDA, one of the most commonly used approaches in app classification tasks [33, 27, 43]. Our results show that word embeddings models were more successful in identifying correct app categories than topic models (LDA). This can be explained based on the observation that the topics generated for our data were of poor quality. Table 2.8 shows the five generated topics with their most probable words. As the results show, the generated topics failed to capture any of the expert-generated categories. For example, while the second topic generated for our educational apps included words such as *game* and *fun*, it failed to represent a coherent category due to the presence of important words from other categories, such as *learn* and *child*. Topics generated for the set of Health apps seem to be more cohesive. For

example, Topic 5 includes words such as *workout*, *exercise*, *weight*, and *fit* which are indicative of a fitness app. Similarly, Topic 6 includes words such as *day*, *medical*, *track*, and *time* which indicate a patient management app. However, both topics also share words with other less cohesive topics, such as Topic 1 and Topic 3, leading the classifier to make inaccurate predictions. In other words, due to the overlapping nature of the different topic categories, the classes are not separable by LDA.

The poor results of LDA can be partially explained based on the limited length of app descriptions [110]. Prior evidence has shown that LDA does not perform well when the input documents are short in length [111, 78]. Specifically, LDA is a data-intensive technique that requires large quantities of text to generate meaningful topic distributions. However, due to the limited nature of description text, applying standard LDA to such data often produces incoherent and overlapping topics [78]. One instance of LDA misclassification in our dataset is the app of the International Journal of Psychology (IJP). This app keeps track of the research in the field of psychology. The description of this app includes statements such as:

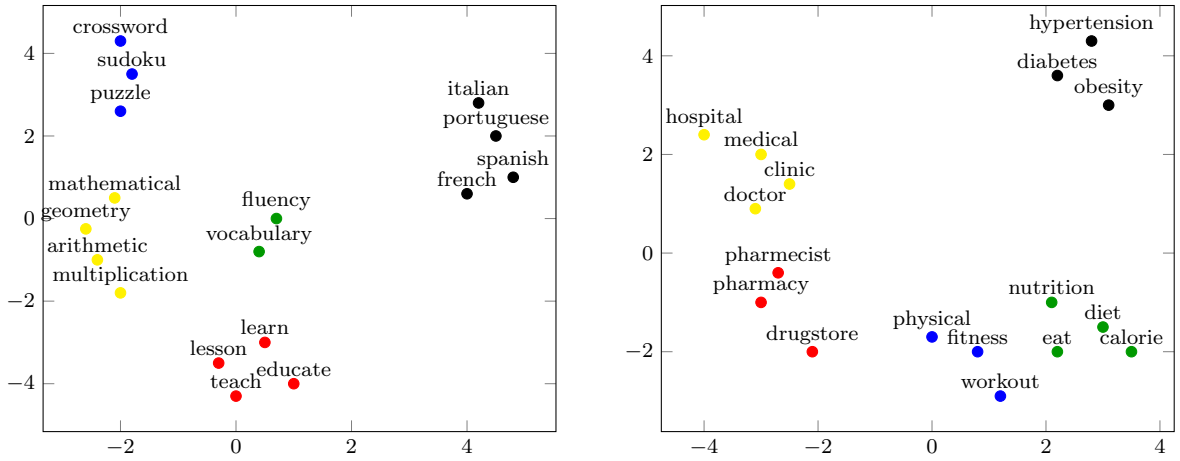
- *Stay current with the latest articles through early view*
- *Receive alerts when new issues are available (opt in)*
- *Save your favorite articles for quick and easy access, including offline*
- *Dynamic references show references in context*
- *Share article abstract and link via email*
- *Access your personal or institutional subscription to IJP on your ipad*

Table 2.8: Topics generated by LDA for our dataset of Education and Health apps.

Dataset	**Topics**	**Most probable words**
Education	Topic 1	learn, word, play, child, help, game, letter, lesson, fun, use
	Topic 2	math, game, level, time, word, kid, child, puzzle, fun, use
	Topic 3	color, kid, book, child, game, school, story, learn, feature, fun
	Topic 4	dictionary, english, use, question, access, period, record, exam, free, time
	Topic 5	school, inform, use, feature, student, english, access, news, include, event
Health& Medical	Topic 1	view, class, schedule, download, today, time, inform, contact, location, exam
	Topic 2	inform, use, injury, patient, provide, help, assess, view, detail, product
	Topic 3	workout, exercise, weight, account, purchase, fit, period, program, hour, renew
	Topic 4	patient, meditation, health, help, inform, day, education, treatment, track, time
	Topic 5	calculate, workout, health, exercise, nutrition, rate, calorie, food, heart, pain
	Topic 6	medicine, patient, doctor, care, medical, prescript, time, help, use, access

IJP connects users to a database of articles related to psychology. This apps perfectly fits into the content-based app category. IJP's description contains words such as *journal*, *article*, *reference*, *research*, *report*, and *study*. These words are semantically related to the groups of words that convey the concept of a database of information. The word embeddings models used in our analysis correctly classified this app. LDA, in contrast, misclassified this app as a function-based app. This happened due to the presence of words such as *access*, *include*, and *use* in IJP's description. These words led to classifying the app to Topic 5 which is mostly related to function-based apps.

Word embedding models tend to be immune to the limitations of topic modeling methods. For instance, Fig. 2.8-a and b show a 2D projection of the GloVe_{300} embeddings (vectors) of a collection of words sampled from the descriptions of our education and health apps. The projection shows that related words tend to appear in separate clusters in the 2D space. For example, words such as *Italian, Portuguese, Spanish*, and *French* which are indicative of foreign languages appeared in a single well-defined cluster (closer in the space). The same applies to the words *crosswords, sudoku*, and *puzzle*, which are



(a) Education **(b) Health&Medical**

Figure 2.8: Example word vectors represented in a 2D space generated by applying PCA on 300-dimensional GloVe word embeddings.

indicative of educational games. Similar patterns of related word clusters can also be observed in the Health category, where words such as *diabetes, hypertension*, and *obesity* appeared near each other in the vector space. This kind of representation provided sufficient information for our classifiers to make accurate predictions.

2.7.2. Word Embeddings vs. Bag-of-Words

Our results also show that bag-of-words methods such as VSM and BM25 can be heavily disadvantaged by the vocabulary mismatch problem of app descriptions. In particular, both BM25 and VSM vectorize app descriptions based on the *TF.IDF* scores of their individual words. In comparison, word embeddings capture the semantic meaning of words during vectorization. In app description classification tasks, the semantic meaning of description words generates more information than their TF.IDF scores. For instance, apps that connect patients with their doctors use different vocabulary to convey this functionality, such as *"connect patients with doctors"* and *"link patients with physician"*. According to BM25 and VSM, the similarity of these two sentences are 8%, and 20%, respectively. In

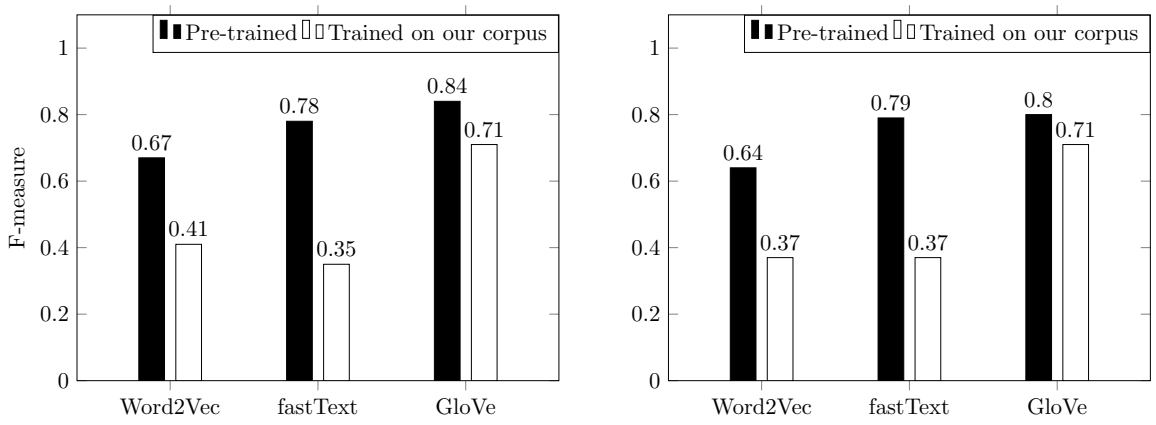
comparison, GloVe captures the 86% similarity between these two sentences with as the vectors of *connect* and *link*, and *doctor* and *physician* appear very close in the space.

2.7.3. GloVe vs. Word2Vec and fastText

Our results show that GloVe has outperformed Word2Vec and fastText (Table 2.5). In general, context-free word embeddings models, such as Glove, Word2Vec, and fastText are known to achieve comparable results. However, their performance can slightly vary depending on the intrinsic complexity of the text corpus [38, 112, 113, 58]. A potential reason for GloVe's better performance is the fact that the vocabulary used to train the model was more comprehensive than the vocabulary used to train the Word2Vec model. In particular, the pre-trained Word2Vec model used in our analysis did not include 1,422 words of the words that appeared in the descriptions of our apps, while 822 words (496 unique words) were missing from the pre-trained GloVe model. Furthermore, the majority of missing vocabulary in GloVe included insignificant words, such as apps names (i.e. *Accelastudy, Fortville, Kidomy, Vuga, etc.*), typos (i.e. *againsttheclock, comapany, jjust, uesd. etc.*), compound names (i.e. *cross-contamination, x-rays, custom-made, cutting-edge*), and unknown English words (i.e. *woao, yorinks, dixio, cassanea*). As for fastText, the embeddings generated for rare words (character n-grams) did not help the classification accuracy as such words were highly uncommon in our dataset.

2.7.4. Pre-trained vs. Locally trained models

In addition to pre-trained word embeddings, we used the Gensim library in Python to train our word embedding models on our corpus of 1,479,203 app descriptions (the entire population of mobile apps collected from the Apple App Store). The corpus was ini-



(a) Education **(b) Health&Medical**

Figure 2.9: The results of classifying educational health-related apps using SVMs using two different word embeddings: pre-trained and trained on our own corpus of apps descriptions.

tially preprocessed by removing non-ASCII characters and URLs. We then trained our different word embeddings models (Word2Vec, fastText, and GloVe) on the corpus. These models were then used to vectorize apps in our dataset.

As Fig. 2.9 shows, for all three embedding models (Word2Vec, fastText, and GloVe), the pre-trained models outperformed the models trained on our corpus. The poor performance of these models can be related to the relatively small size of the corpus. In order to achieve the most semantically meaningful vector representations of words, word embeddings models need a rich corpus where common English words are significantly more common than rare words. Our corpus of app descriptions contains numerous rare words that are not common English words, thus, resulting in low-quality semantic representations (embeddings) of words. In particular, in our dataset of mobile apps descriptions, 55% of the words appeared less than four times in the corpus.

Table 2.9: The average running time (in seconds) of the different vectorization methods.

Approach	Load Model	Extract Features	Classification	Total
Baseline	-	6.92	19.64	26.56
VSM	-	4.41	1.26	5.67
LDA	13.76	0.31	0.42	14.49
Word2Vec	113.27	206.77	1.9	321.94
GloVe	154.39	23.48	1.9	179.77
fastText	111.88	4.65	1.9	118.43

2.7.5. Time Analysis

In terms of running time, word embedding models typically require more time than other techniques such as LDA and VSM in order to classify mobile apps. Table 2.9 shows the execution time of different vectorization methods when classifying educational apps. Time was measured on an Intel(R) Core(TM) i7-7500U CPU 2.7 GHz, with 12.0GB of RAM. On average, VSM is the fastest method since it only requires the calculation of TF.IDF values of words. BM25 requires slightly more time since feature selection is applied for each training set. Word embeddings are on average slower in generating the classification results since loading a model can be a time-consuming task. However, once the embedding model is loaded, feature extraction and app classification require approximately the same amount of time as other methods to be completed.

2.8. Threats to Validity

The study presented in this chapter has several limitations that could potentially limit the validity of the results. In what follows, we discuss these threats along with our mitigation strategies.

2.8.1. Internal Validity

Internal validity refers to confounding factors that might affect the causal relations established throughout the experiment [64]. A potential threat to the proposed study's internal validity might stem from the fact that human judgment was used to classify our apps and create our ground-truth. Different judges might classify the data differently, which might impact the results of our automated classifiers. Despite these subjectivity concerns, it is not uncommon in text classification tasks to use humans' judgment to prepare the ground-truth. Therefore, these threats are inevitable; however, they can be partially mitigated. For instance, in our analysis, this threat was mitigated by using multiple judges and majority voting and by utilizing pre-existing oracles that were proposed by domain experts [28, 29]. Similarly, a threat may arise from the fact that the categories for our validation set of Sharing Economy apps were generated by the authors. Nonetheless, the authors have published multiple papers on the Sharing Economy [114, 87, 115] and received multiple federal funding grants to develop accessible Sharing Economy solutions for their local community, thus, they can be considered as domain experts in the field.

Other internal validity issues might arise from the specific word-embedding methods, classification algorithms, and open source tools (Scikit-learn) used in our analysis. For example, we used GloVe, Word2Vec, and fastText to generate our word-embeddings. Other techniques, such as BERT, and other types of classification algorithms, such as Hierarchical Agglomerative Clustering (HAC), might arrive at different results.

2.8.2. External Validity

Threats to external validity impact the generalizability of the results obtained in the study [64]. In particular, the results of our experiment might not generalize beyond the specific experimental settings used in this chapter. External validity concerns might be raised about the fact that only 600 apps sampled from two application domains were considered in our analysis, thus, the results of our empirical investigation might not generalize to other apps or domains. To mitigate this threat, we uniformly sampled our apps from the collection of all the educational and health-related apps in the Apple App Store [63]. This helped us to mitigate sampling problems and increase the confidence in our results. To further enhance the generalizability of the results, we validated our classification model on a third dataset sampled from the domain of Sharing Economy. The results came out aligned with our results on other datasets, providing evidence on the applicability of our approach to other application domains.

2.8.3. Construct Validity

Construct validity is the degree to which the various performance measures accurately capture the concepts they intend to measure. In our experiment, there were minimal threats to construct validity as the standard performance measures (recall, precision, and the F-measure), which are extensively used in related research, were used to assess the performance of our different investigated methods. We believe that these measures sufficiently captured and quantified the different aspects of performance we were interested in.

2.8.4. Conclusion Validity

Conclusion validity is concerned with issues that might affect the ability to draw the right conclusion about the relations between the treatment and the outcomes of the experiment [64]. To control for such threats, our data were tested for normality prior to our analysis and appropriate non-parametric statistical tests were then used to measure the difference in performance among the different treatments (classification settings). Overall, we were able to reject our null hypotheses with high statistical power. Furthermore, in our human experiment, our subject sample included 12 participants of female and male subjects (age between 21 - 54) and with various levels of experience in the Sharing Economy as service providers and receivers. We applied randomization whenever possible to minimize any confounding effects.

2.9. Conclusion and Future Work

In this chapter, we proposed a new approach for classifying mobile apps based on their app store descriptions. Our approach utilized models of word embeddings to generate numeric semantic representations of app descriptions. These vector representations were then classified to produce more cohesive categories of mobile apps. The performance of our approach was evaluated using a dataset of apps sampled from the Education, Health&Fitness, and Medical categories of the Apple App Store. Expert-generated categorizations of these apps were used to produce our ground-truth. Our results showed that word embeddings produced using the pre-trained GloVe_{300} led to higher quality categorization than embeddings generated using Word2Vec and fastText. Our results also showed that word embeddings were able to outperform other vectorization techniques such

as bag-of-words (VSM and BM25) and topic modeling (LDA) and other baselines which considered app meta-data attributes as classification features [31]. To further validate our results, we applied our GolVe classification model on a third dataset of Sharing Economy apps. The results showed that our model was able to achieve accuracy levels comparable to the accuracy achieved on the first two datasets. We further ran a study with 12 participants to assess the quality of our classifier. The results showed that our study participants classified our apps with a high degree of agreement with our approach.

Our work in this chapter is expected to help users discover apps that match their specific interests more effectively. Developers can also use our approach to identify their direct competition in the app store. In terms of future work, our analysis in this chapter can be extended along three main directions:

- **More data:** More analysis, utilizing more expert-generated categorizations of apps across a broad range of application domains will be conducted. Our objective is to determine a global set of configuration settings that can be used to dynamically generate more accessible categorizations of apps.

- **Tool support:** A working prototype will be developed to implement our findings. The prototype will be ideally implemented in a mobile app with a user-friendly interface to aid mobile app users in finding apps that meet their specific needs.

- **Extrinsic evaluation:** Our evaluation in this chapter was mainly intrinsic, based on how well the generated categories correlated with existing classifications. While such an evaluation can be sufficient for model assessment, it does not capture the

practical significance of the results. Therefore, a main direction of future work will be dedicated to extrinsic evaluation. 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 prototype will be provided to selected groups of stakeholders, such as health professionals, educators, and app developers to be used as an integral part of their app search and development activities. Evaluation data will be collected through surveys that will measure the approach's usability, scalability, and overall value to users.

Chapter 3. Privacy: User Perspective

The proliferation of mobile applications (app) over the past decade has imposed unprecedented challenges on end-users privacy. Apps constantly demand access to sensitive user information in exchange for more personalized services. These—mostly unjustifiable—data collection tactics have raised major privacy concerns among mobile app users. Such concerns are commonly expressed in mobile app reviews, however, they are typically overshadowed by more generic categories of user feedback, such as app reliability and usability. This makes extracting user privacy concerns manually, or even using automated tools, a challenging and time-consuming task. To address these challenges, in this chapter, we propose an effective unsupervised approach for summarizing user privacy concerns in mobile app reviews. Our analysis is conducted using a dataset of 2.6 million app reviews sampled from three different application domains. The results show that users in different application domains express their privacy concerns using domain-specific vocabulary. This domain knowledge can be leveraged to help unsupervised automated text summarization algorithms to generate concise and comprehensive summaries of privacy concerns in app review collections. Our analysis is intended to help app developers quickly and accurately identify the most critical privacy concerns in their domain of operation, and ultimately, alter their data collection practices to address these concerns.

3.1. Introduction

Mobile apps are designed with a set of user goals in mind. A user goal can be described as any abstract objective that the system under consideration should achieve [116].

This chapter was previously published as Ebrahimi, Fahimeh, and Anas Mahmoud. "Unsupervised Summarization of Privacy Concerns in Mobile Application Reviews." 37th IEEE/ACM International Conference on Automated Software Engineering. 2022.

For example, the goal of Sharing Economy apps (e.g., Uber and Airbnb) is to foster social capital and economic growth in resource-constrained communities [117, 101] while the goal of Health&Fitness apps is to promote healthy lifestyles among children and adults [118, 51]. However, driven by fierce market competition, app developers frequently deviate from their original goals. These deviations often come in the form of extreme privacy-invading tactics, such as constant location-tracking [6], unsolicited data collection [5, 7], or features that are intentionally engineered to lure users into sacrificing their privacy in exchange for more personalized experiences [8, 9].

Apps that do not adequately address their users' privacy concerns are often deemed untrustworthy or even abandoned by their users [119, 120, 121]. Therefore, in order for apps to survive market selection, app developers must constantly monitor their users' feedback and adjust their data collection strategies accordingly [16]. Users commonly communicate their feedback with app developers through textual app reviews [10, 122, 123, 15]. General-purpose review mining techniques, such as text classification and topic modeling, have been extensively used to classify such feedback into different types of actionable software maintenance requests [124, 86, 125, 126, 85, 127]. However, due to their sparsity and domain dependency, privacy concerns are frequently misclassified or under-recognized by these techniques [15]. For example, users of ridesharing apps (e.g. Uber and Lyft) might complain about the constant tracking of their location, while users of investing apps (e.g., Robinhood and Coinbase) might raise concerns about sharing their social security or bank information with the app. Such domain-specific feedback can be easily missed in the presence of more dominant categories of technical concerns. Consequently, a *one-size-fits-all* approach may not be suitable for detecting privacy concerns across all domains.

To address these challenges, in this chapter, we propose a new unsupervised approach for summarizing privacy concerns in the mobile app market. Our approach is based on the assumption that privacy concerns are domain-specific. Therefore, leveraging the vocabulary that is commonly used by app users to express their privacy concerns in a specific domain can help generic text summarization algorithms generate more concise and representative summaries of these concerns. Our approach is evaluated using a large dataset of user reviews sampled from the domains of mental health, investing, and food delivery apps. Our long-term goal is to help app developers identify the critical privacy concerns in their domain of operation, alter their data collection practices to mitigate these concerns, and ultimately, survive user selection.

The remainder of this chapter is organized as follows. In Section 3.2, we motivate our research. In Section 3.3, we describe our subject domains and experimental dataset. In Section 3.4, we present our procedure for extracting privacy keywords from app reviews. In Section 3.5, we propose a novel algorithm for summarizing privacy-related mobile app reviews. In Section 3.6, we discuss our key findings. In Section 3.7, we address the main limitations of the study. Finally, in Section 3.8, we conclude the chapter and describe our directions of future work.

3.2. Background and Motivation

Privacy in the mobile app market has received significant attention from the research community over the past decade. However, the majority of existing literature is focused on detecting privacy policy violations and preventing data leaks, while less attention has been paid to mining end-users' privacy concerns [10].

Earlier evidence on extracting privacy concerns in the mobile app market can be found in Khalid et al. [16]. The authors manually examined and classified thousands of one and two-star app reviews to get a better sense of end-users' complaints and their impact on app ratings. The analysis revealed that reviews including complaints about privacy-invading practices were often associated with the most negative impact on ratings. In another study, McIlroy et al.'s [15] qualitative analysis of 7,000 user reviews revealed that close to 17% of examined reviews raised privacy concerns. These concerns were expressed using more varied language than other types of technical issues.

Ciurumelea et al. [123] used iterative content analysis to develop a taxonomy of actionable issues in mobile app reviews, including compatibility, usage, resources, pricing, and privacy. In a more recent work, Hatamian et al. [122], proposed MARS, a tool for summarizing privacy-related mobile app reviews and classifying them into a set of predefined security threats, including spyware, phishing, and spam. Informative reviews were detected based on a keyword catalog seeded with the initial keywords: *privacy* and *security*. This catalog was iteratively expanded with more privacy-related keywords using word frequency analysis. Extracted keywords were then used to tune different text classifiers. MARS was able to classify 2,412 privacy-related reviews with a recall and precision of 91.30% and 94.84% respectively.

Besmer et al. [128] analyzed a massive dataset of mobile app reviews collected over the period of four years. The results showed that reviews that contained complaints about app privacy had lower star ratings and more negative sentiment than other reviews. The results also showed that users found privacy-related reviews to be more helpful than others. In another study, Mukherjee et al. [129] identified privacy-related app reviews using a

generic set of privacy-related keywords. The authors found that only 0.5% of reviews were related to end-user security and privacy. Nguyen et al. [130] also used a set of 102 generic keywords to extract potential security and privacy concerns from 2,583 Google Play apps. A manual analysis of 4,000 reviews of these apps showed that 14% of them were either privacy or security-related. The authors also reported that preceding privacy reviews were a significant factor in predicting privacy-related app updates.

In summary, the majority of existing work employs generic text classification and NLP-based methods for detecting privacy concerns in mobile app reviews [131]. However, the results largely indicate that these generic solutions often struggle to capture domain-specific privacy concerns. For instance, supervised classification techniques rely on the presence of manually generated ground-truth datasets. Thus, these techniques are constrained to a single rubric of predefined categories [127, 126, 85]. Consequently, specific categories related to user privacy can be easily missed in the ground truth data. Unsupervised topic modeling techniques, such as Latent Dirichlet Allocation (LDA) [42], have also been applied to extract privacy information from app store reviews [132]. However, such techniques do not perform well with small, unstructured, and semantically-restricted text such as user reviews [133, 111, 78, 134].

Existing work has also revealed that users often express their privacy concerns using more varied language than other technical issues [15]. Consider for example the following three reviews selected from the domains of mental health, investing, and food delivery apps. The word *Facebook* clearly indicates a privacy concern in the mental health domain. However, in food delivery, the same word is used to express a customer support problem, while in the investing domain the word is used to express an user registration issue.

- **Mental Health**: "*Won't even let me sign up after collecting all of my Facebook data, just stole my identity.*"
- **Investing**: "*I got zero response back. I even blasted their Facebook but got nothing.*"
- **Food Delivery**: "*It doesn't recognize my facebook account so I can't even register for this.*"

Motivated by the limitations of existing work, in this chapter, we propose an unsupervised approach for summarizing user privacy concerns in the mobile app market. We initially describe a systematic method for extracting privacy-related vocabulary from three different application domains. Extracted vocabulary is then leveraged to generate concise and comprehensive summaries of privacy concerns in app reviews.

3.3. Data Collection

Our underlying assumption in this chapter is that app users express their privacy concerns using different terminology that is directly related to their apps' specific functionality. To verify this assumption, we collected a large-scale dataset of user reviews from three different application domains: mental health, food delivery, and investing. Investing apps have become increasingly popular in recent years due to the increasing interest in cryptocurrency trading. Zero-commission trading fees and continuous media coverage have brought in millions of new first-time traders. For example, one of the most popular investing apps, *Robinhood*, reported that close to 6 million new users joined the platform in 2021 [135]. Similarly, the domain of food delivery has experienced massive growth during the past two years. In particular, the demand for food delivery services has significantly increased due to the COVID-19 pandemic. For example, the three major food

delivery apps - DoorDash, UberEats, and GrubHub have all reported a significant increase in revenue generated during the lock-down order of 2020 [136]. This global health crisis has also led to a significant spike in the number of active users of mental health apps. People frequently resorted to these apps as a safer and inexpensive alternative to help them cope with the consequences of social isolation, unemployment, and economic hardships [137, 138].

To collect reviews from these three different domains, we identified the top-100 apps in the categories of Finance (investing), Food&Drink (food delivery), and Health&Fittness (mental health) on Google Play and the Apple App Store. Apps that met the following criteria were included in our dataset:

- For an app to be included in our analysis, we only considered apps with 10,000 reviews or more. This number of reviews was necessary to include only popular and well-established apps. We lowered this number to 5,000 for mental health apps as apps in this category do not get as many reviews.

- For the Finance category, banking *"all-in-one"* apps were excluded as the majority of these apps did not provide investing services. For Food&Drink, specific restaurant delivery apps, such as *Papa John's Pizza & Delivery* official app, were also excluded as they did not operate as independent delivery services. In the Health&Fitness category, physical health apps that did not explicitly support mental health were excluded.

After examining the top-100 apps in each category, eight investing, five mental health, and five food delivery apps were included. For each of these apps, we collected

all textual reviews available on the Apple App Store and Google Play using Python web scrapers. Overall, 696,073, 1,708,831, and 204,374 reviews were collected for our set of investing, food delivery, and mental health apps respectively. The distribution of these reviews over apps is shown in Table 3.1.

3.4. Extracting Privacy Vocabulary

In this section, we empirically examine our assumption that privacy concerns in mobile app reviews are expressed using domain-specific vocabulary.

3.4.1. Privacy Term Extraction

Our analysis is conducted over low-rating (one and two stars) reviews in our dataset. These reviews are more likely to contain user complaints or useful feedback than high-rating reviews [15, 16, 139]. In total, 385,951, 511,032, and 43,647 reviews from the domains of investing, food delivery, and mental health are included in our analysis. Table 3.1 shows the total number of 1-2 star reviews for each app in our dataset.

Fig. 3.1 describes our indicator term (keyword) extraction process. The goal of our analysis is to generate a catalog of terms and phrases that signal privacy-related issues in different application domains. To generate such a catalog, we follow a systematic iterative process of word generation. We begin our analysis by seeding the catalog with the words *privacy*, *private*, and *security* [130]. These words are then used as search queries to locate potential privacy-related reviews in our dataset. The first iteration of the search returned 187, 753, and 629 reviews for the apps in the mental health, investing, and food delivery domains respectively. Each review is then manually examined by three judges to locate any other keywords (unigrams or bigrams) that are likely to be indicative of privacy is-

Table 3.1: The number of user reviews extracted (1-2 stars) for each app in our dataset.

Investing		**Food Delivery**		**Mental Health**	
App	**Reviews**	**App**	**Reviews**	**App**	**Reviews**
Robinhood	451,016 (325,534)	UberEats	748,584 (265,713)	Calm	106,181 (22,983)
Acron	76,761 (15,954)	DoorDash	598,513 (122,857)	Headspace	78,989 (16,376)
Stash	40,385 (10,683)	GrubHub	223,566 (63,776)	Sanvello	8,554 (698)
ETrade	15,807 (9,297)	Postmates	107,564 (53,579)	Talkspace	5,054 (2,928)
Fidelity	50,224 (9,034)	Seamless	30,604 (5,107)	Shine	5,596 (662)
TD Ameritrade	30,369 (8,973)				
Schwab	14,988 (4,596)				
Personal Capital	16,523 (1,880)				
Total	696,073 (385,951)	**Total**	1,708,831 (511,032)	**Total**	204,374 (43,647)

sues. Each judge has to answer the questions: *does this review raise any form of privacy concern?* And if so, *what keyword(s) in the review are indicative of such concerns?* All judges hold professional degrees in software engineering as well as have an average of 6+ years of experience in app development. A pilot labeling session was held before running the actual analysis to explain the process and address any concerns. Terms generated after the first round are then used to retrieve the second set of reviews. Basically, any review that contained any of the identified keywords and did not appear in previous searches is included in the search results. This process is repeated until saturation, or no more new keywords are found.

Notice that several keywords retrieved a large number of matches. For instance, the word *location* returned 10,829 reviews for the apps in the food delivery domain. Examining such a large number of reviews manually can be an exhaustive and error-prone task. Therefore, for such large sets, we only examine a statistically representative stratified sample of reviews. A sample size of 300 reviews is sufficient to maintain at least a 95% confidence level. Among the identified keywords, only nine of the keywords retrieved more

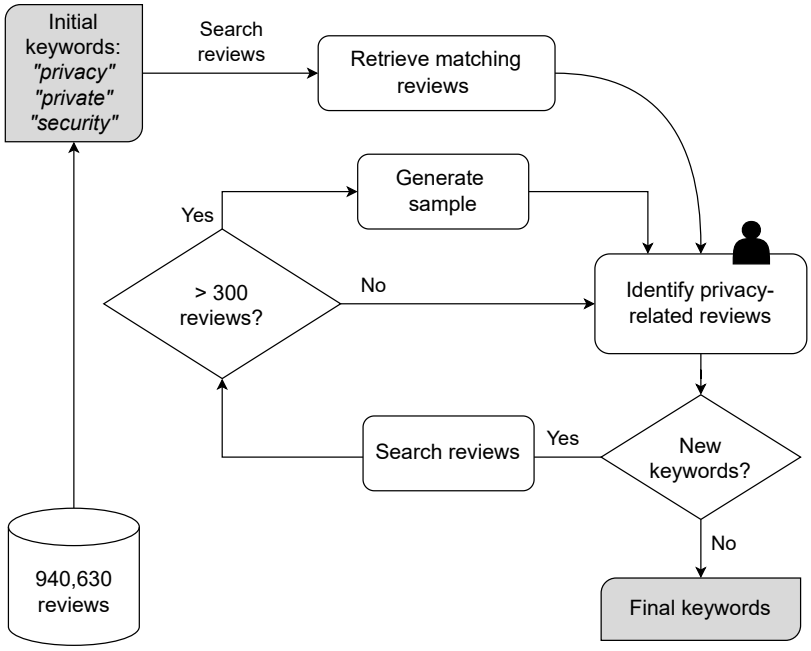


Figure 3.1: Our privacy keyword extraction procedure.

than 300 reviews and needed the sampling phase. At the end of this process, 26 unique privacy-related keywords or phrases (e.g. *personal info*) were extracted from our dataset. These keywords are listed in Table 3.2. The table also shows the total number of reviews retrieved in each domain as well as the number of reviews that are privacy-related.

3.4.2. Keyword Analysis

Our keywords extraction process generated 26 unique privacy-related keywords. In Table 3.3, we show the percentage of privacy-related reviews that are retrieved by each of our keywords in each application domain. We use Chi-square (χ^2) to test for statistical significance in these results. Our null hypothesis (H_0) is that there is no difference in the percentage of privacy-related reviews retrieved by each privacy keyword between all domains. The alternative hypothesis (H_1) is in favor of the dependency between the do-

Table 3.2: The results of our indicator keyword extraction process, showing the number of reviews retrieved by each keyword at each round in each domain along with the number of privacy-related reviews (shown in parenthesis) identified.

Round	Keywords	Mental Health	Investing	Food Delivery
1st	*privacy*, *private*, *security*	187 (**140**)	753 (**291**)	629 (**183**)
2nd	*personal info*, *permission*, *user data*, *facebook*, *patient info*, *bank statement*, *bank login*, *credit card*, *SSN*, *social security*	869 (**200**)	1,805 (**586**)	1,402 (**289**)
3rd	*camera*, *microphone*, *GPS*, *location*, *job history*, *birth*	45 (**11**)	595 (**165**)	905 (**39**)
4th	*driver license*, *real name*, *last name*, *imei*, *identification info*, *email*	303 (**40**)	366 (**16**)	359 (**4**)
Total		1,404 (**391**)	3,519 (**1,058**)	3,295 (**515**)

main and the keywords, in other words, the recall of different keywords when retrieving privacy-related reviews is significantly dependent on the domain. Since we have two variables (privacy-related and non-privacy-related) and three groups (domains), the degree of freedom of our test is set to $2 = (2-1)*(3-1)$. Given this degree of freedom and the confidence levels of 0.001, 0.01, and 0.05, χ^2 critical values are set to 13.816, 9.210, and 5.991, respectively. H_0 will be rejected if the χ^2 value is larger than the critical values. The last column of Table 3.3 shows the chi-square test results and the confidence level for each keyword. The results show that, for the majority of domain-specific keywords, we can reject H_0 with a confidence level of at least 0.05.

The results also show that the number of privacy-related reviews retrieved by the generic keywords (e.g., *privacy*, *security*, etc.) is not dependent on the application domain. Most of the reviews that contain the keyword *private* are not necessarily privacy-related. For instance, in the food delivery domain, only 23% of the reviews that contain this keyword raise privacy concerns, commonly appearing in reviews such as, *"I live on first floor first apt in a private building."* The same observation holds for other keywords that are

frequently associated with privacy in the literature, such as *camera*, *security*, and *permission*. For example, users in the food delivery domain use the keyword *permission* to complain about their orders being canceled without their permission.

In general, the majority of our identified keywords are domain-specific. For example, the phrase *"last name"* appears as one of the main privacy-indicator terms in the mental health domain. All reviews (100%) that contain *"last name"* in this domain are privacy-related. However, in the food delivery and investing domains, less than 10% of reviews that contain *"last name"* are privacy-related. We also observe that some keywords are more domain-specific than others. For example, the term *"driver's license"* appears frequently in the privacy-related reviews of investing apps. Users are mainly complaining about apps asking for photocopies of their driver's license, such as, *"why do they need pictures of both sides of my driver's license, they already verified my bank account and who do they share this information with?"* However, none of the reviews that contain this keyword in the food delivery domain is privacy-related. In general, customers of food delivery apps use *"driver's license"* to ask questions about working for the app, such as, *"Can I work for this app if my driver's license was out of state?"* or complain about drivers, such as, *"I don't think my delivery kid has a driver's license."*

The results also show that some keywords are good signals of privacy issues only in two domains, but are mainly associated with false positives in the third domain. For example, the keyword *GPS* is almost always associated with privacy-related issues in investing and mental health domains. Users in these domains frequently use this keyword to express concerns about apps tracking their location. However, in the food delivery domain, users have no issue with delivery apps demanding access to their location to get their food

Table 3.3: The percentage of privacy-related reviews in the reviews retrieved by each keyword in each application domain. The significance of word × domain dependency is measured using Chi-square (χ^2), $^{*}p \leq 0.05$, $^{**}p \leq 0.01$, $^{***}p \leq 0.001$

Keyword	**Invest.**	**Mntl. Health**	**Food Deliv.**	χ^2
microphone	93%	0%	40%	15.76***
social security / SSN	52%	14%	15%	13.68**
identification info	100%	0%	25%	6.36*
bank login	11%	0%	0%	37.35***
birth	26%	9%	0%	82.67*
camera	23%	0%	11%	15.38*
driver license	35%	0%	0%	1.5
credit card	4%	1%	3%	6.25*
email	2%	13%	0%	59.11***
facebook	15%	23%	2%	29.55***
last name	10%	100%	4%	6.54*
private	30%	55%	23%	15.8***
job history	0%	100%	0%	-
imei	0%	100%	0%	-
patient info	0%	100%	0%	-
location	29%	44%	0%	102.73***
GPS	100%	100%	1%	132.66***
privacy	94%	98%	67%	4.93
personal info	71%	100%	94%	5.18
user data	100%	100%	50%	0.84
security	19%	19%	7%	1.92
permission	4%	27%	3%	3.18

delivered to their address. In general, despite their very common occurrence in the reviews of food delivery apps, keywords such as *GPS* and *location* are not indicative of privacy concerns in this domain, mainly appearing in reviews such as, *"only two restaurants for my location. horrible!"* or *"the driver needs to update their GPS to the new map."*

3.4.3. Keyword Co-occurrence Analysis

We observed while labeling the data that domain-specific keywords tend to co-occur in privacy-related reviews. For example, in the review *"it asks for your first and last name, email address and facebook account. I did not feel comfortable sharing personal*

information", a privacy concern of using a particular mental health app is expressed using four indicator terms, *"last name", "email", "facebook",* and *"personal info"*. To further examine this observation, we calculate the Normalized Pointwise Mutual Information (NPMI) between the set of privacy indicator terms extracted from our dataset. NPMI is an information-theoretic measure of information overlap, or statistical dependence, between two words [140]. Formally, NPMI between two words w_1 and w_2 can be measured as the probability of them occurring in the same text versus their probabilities of occurring separately. Assuming the collection contains N reviews, PMI can be calculated as:

$$NPMI(w_1, w_2) = \frac{\log_2(\frac{\frac{C(w_1,w_2)}{N}}{\frac{C(w_1)}{N}\frac{C(w_2)}{N}})}{-\frac{C(w_1,w_2)}{N}} = \frac{\log_2(\frac{P(w_1,w_2)}{P(w_1)P(w_2)})}{-P(w_1,w_2)} \tag{3.1}$$

where $C(w_1, w_2)$ is the number of reviews containing both w_1 and w_2, and $C(w_1)$ and $C(w_2)$ are the numbers of reviews containing w_1 and w_2 respectively. NPMI is normalized using the negative log-transformed count of the number of times w_1 and w_2 appear together. If w_1 and w_2 are frequently associated, the probability of observing them together will be much larger than the chance of observing them independently. This results in NPMI > 0. If there is 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 (NPMI < 0).

The results of our NPMI co-occurrence analysis are shown in Fig. 3.2. The figure shows the semantic distance between our keywords projected on a 2D map. In the mental health domain, *"email"*, *"last name"*, *"GPS"*, and *"Facebook"* commonly co-occur together in privacy reviews. These keywords retrieved the highest percentage of privacy-related reviews in the mental health domain (Table 3.3). Similarly, the keywords *"SSN"*, and *"bank*

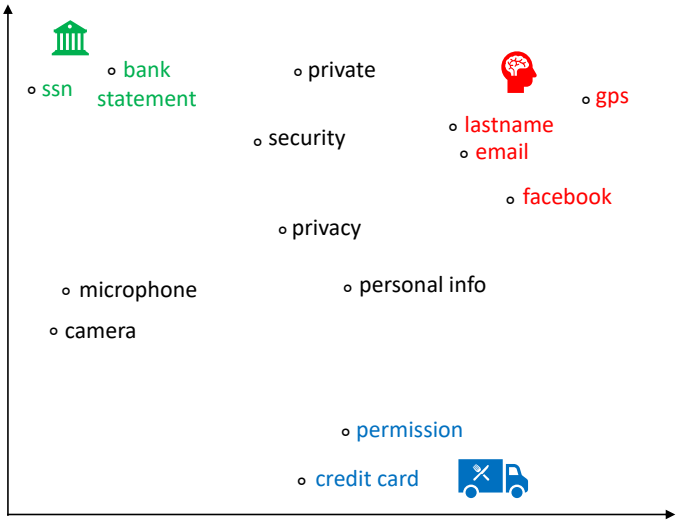


Figure 3.2: The NPMI distance between extracted privacy-related keywords from our different application domains.

statement" commonly co-occur in the privacy reviews of investing apps. We further observe that keywords from different application domains seem to be standing at the same semantic distance from the seed *"privacy"*.

In summary, our analysis in this section provides evidence that the keywords used by app users to express their privacy concerns are domain-specific. Such keywords are largely derived from the features of the app and its operational characteristics. While some keywords may provide a strong signal of privacy concerns in one domain, they may not be indicative of privacy issues in other domains. We also observe that domain-specific privacy-indicative keywords tend to frequently co-occur together in app reviews. In the next section, we show how such insights can be leveraged to generate concise summaries of privacy concerns in mobile app review collections.

3.5. Review Summarization

The first phase of our analysis has revealed that users' privacy concerns in mobile app reviews are commonly expressed using domain-specific vocabulary. Therefore, a *one-size-fits-all* approach for detecting privacy concerns across different application domains in the app store is destined to fail. This problem is further aggravated by the fact that privacy concerns are sparse, only appear in a small percentage of reviews, and are frequently overshadowed by more dominant categories of user feedback, such as concerns about the app's reliability (e.g., reporting bugs) or usability (e.g., requesting features) [129, 128, 16]. These limitations hinder the ability of supervised learning techniques to detect privacy-related reviews as sparse categories of data can be easily missed in the training dataset. The problem also severely affects unsupervised topic modeling techniques, such as LDA, where generated topics are naturally representative of dominant themes in the data [141].

To work around these limitations, we propose and evaluate an unsupervised domain-specific approach for summarizing privacy concerns in the mobile app store. Our approach leverages domain knowledge to point our summarization approach toward the most prevalent privacy concerns in user reviews. Timely detection and handling of such concerns can be critical for app survival as recent evidence has shown that privacy-related reviews are commonly accompanied by negative sentiment and low ratings [128, 16, 142].

3.5.1. Text Summarization

The goal of summarization techniques is to capture the underlying dominant themes in a text corpus (source text) and represent them as cohesively and concisely as possible, in other words, to generate meaningful summaries [143, 144]. In the context of

app reviews, automated summarization techniques are used to assimilate the perspectives of a large number of users to bring app developers' attention to any pressing issues that need to be addressed in future releases [145]. Summarization techniques can be either extractive or abstractive. Abstractive techniques construct novel descriptions of the main ideas in a source text. Extractive techniques, however, group together specific key sentences from the source text to generate a concise summary of the text. Abstractive techniques are commonly known to be more sophisticated as lexical parsing and paraphrasing are needed to generate novel and meaningful summaries [146]. Therefore, they perform better when applied to semantically rich text, such as scientific documents or news articles. However, user reviews are short and often expressed using informal and semantically restricted jargon [147, 148], making extractive techniques more effective in this context.

In general, extractive summarization techniques leverage the frequencies of individual words to estimate their importance to the source text [146]. The likelihood that a sentence from the source text will be considered as a representative summary is positively correlated to the average perceived importance of its words [144]. In semantically restricted text collections (e.g., user reviews), where individual text artifacts are short, the importance of words can be determined using a technique such as Hybrid TF.IDF [149]. TF.IDF consists of two components, Term Frequency (TF) and Inverse Document Frequency (IDF). TF is the raw frequency of a word in a document and IDF is an indicator of how much information this word provides. For a collection of short texts, Hybrid TF.IDF modifies the TF of words by dividing their frequency by the number of unique words in the entire text collection (N). This modification is necessary when dealing with

user reviews as the probability of individual words occurring multiple times in a single review is relatively low. Formally, the Hybrid TF.IDF of a word w can be computed as:

$$HybridTF.IDF(w) = \frac{f(w,R)}{\sum f(w,r)} \times \log \frac{|R|}{df(w)} \tag{3.2}$$

where $f(w,R)$ is the term frequency of the word w in all reviews, $\sum f(w,r)$ is the total number of unique words in the review collection, R is the total number of reviews in the collection, and $df(w)$ is the number of reviews that contain w.

Given the above assumptions, Hybrid TF.IDF—as an extractive summarization technique—first calculates the importance of individual reviews as the average of their individual words' Hybrid TF.IDF values. The algorithm then selects the top K most important reviews as summaries. To control for redundancy, before a review r_i is added to the summary, the algorithm makes sure that r_i does not have a textual similarity above a certain threshold with the reviews already in the summary. The textual similarity between two reviews can be calculated as the cosine similarity between their TF.IDF vectors.

3.5.2. Privacy Review Summarization

Extractive summarization techniques work well for online text corpora [89]. However, due to their reliance on words' frequencies, sparse themes in the data tend to be either missed or underrepresented in the summary. To work around these limitations, we alter Hybrid TF.IDF in two ways. First, we adjust the weight of privacy indicator keywords in each domain to the maximum Hybrid TF.IDF value calculated for the domain. For example, after removing English stop-words and performing lemmatization, the maximum Hybrid TF.IDF calculated in the entire set of reviews for the mental health apps in our dataset is equal to 0.022. This weight is assigned to the list of privacy indicator key-

words (Table 3.3) identified for this domain, including *"Facebook", "IMEI", "patient", "location", "email", "GPS", "last name"*, and *"real name"*. The same is applied to the privacy-indicator words identified for the investing domain and the food delivery domain. In addition, the generic catalog seeds (*"privacy", "private"*, and *"security"*) are also assigned to the same maximum weight in each domain.

The second adjustment is related to the way redundancy is controlled in the summary. To minimize the probability of retrieving reviews raising similar concerns, we enforce a similarity threshold calculated using the word embeddings of reviews. For a highly ranked review (based on its average Hybrid TF.IDF score) to be added to the summary, it has to stand at a specific minimum semantic distance from other reviews already in the summary. This distance is calculated in classical Hybrid TF.IDF using the cosine similarity between the TF.IDF vectors of reviews. However, relying on the textual similarity between reviews can lead to information loss. Word embeddings can overcome this problem by relying on the meaning of reviews rather than their lexical structure.

In our analysis, we use GloVe to calculate the word embeddings of individual reviews. GloVe [38] is a popular word embedding model that uses the similarities between words as an invariant to generate their vector representations. In general, word embeddings represent individual words in a corpus using multi-dimensional vectors of numeric values that are derived from the intrinsic statistical properties of the corpus. GloVe initially constructs a high dimensional matrix of words co-occurrence. Dimensionality reduction is then applied to the co-occurrence count matrix of the corpus. By applying a matrix factorization method on the count matrix, a lower dimension matrix is produced, where each row is the vector representation of a word. To conduct our analysis, we converted the

list of pre-processed tokens in each user review into a vector of word embeddings using the pre-trained model of GloVe. We then used the generated word embeddings to represent the review. Word collection embeddings can be computed using operations on word vectors, such as their unweighted averaging/summation [68], Smooth Inverse Frequency (SIF) [69], and Doc2Vec [70, 71]. In our analysis, we used the simple unweighted averaging method to obtain an embedding for each review in our dataset [69, 73]. Averaging word vectors has been proven to be a strong baseline for paragraph representation, especially in cases when the order of words in the text is unimportant [74]. Algorithm 1 describes our privacy-concern summarization approach.

Algorithm 1 A description of our summarization algorithm.

for review r_i in R **do** ▷ R : reviews in the collection
    for word w_j in r_i **do**
        if $w_j \in$ privacy_keywords **then** ▷ is w_j an indicator keyword?
            w_j.weight = max_tf_idf ▷ increase the weight
        else
            w_j.weight = Hybrid.TF.IDF(w_j) ▷ calculate Hybrid.TF.IDF
        r_i.total += w_j.weight ▷ sum up words weights
    r_i.weight = r_i.total / $|r_i|$ ▷ calculate review r_i weight

R' = R.sort(DES) ▷ sort reviews based on their Hybrid TF.IDF
$S = \{r_0\}$ ▷ add the top ranked review to the summary
count = 1 ▷ review length

for r_i in R' **do**
    if GloVe.similarity(r_i, S) ¡ λ **then** ▷ λ: similarity threshold
        S.add(r_i) ▷ Add r_i to summary S
        count++
        if *count* == k **then** ▷ k is desired summary length
            break;

3.5.3. Evaluation

To evaluate our proposed algorithm, we summarize a test dataset of reviews collected from a new set of apps sampled from our three application domains. Using a sep-

Table 3.4: The test dataset used in our analysis.

Domain	App	Avg. Rating	# of Reviews (1-2 star)
Investing	Wealthfront	4.8	4,437 (**320**)
	Stockpile	4.7	4,412 (**999**)
Mental Health	eMoods	4.8	1,774 (**75**)
	Happify	4.5	2,393 (**783**)
Food Delivery	goPuff	4.5	44,397 (**8,145**)
	Delivery.com	4.8	2,661 (**823**)
Total			60,074 (**11,145**)

arate test dataset can help to validate our assumptions regarding the generalizability of our approach over each domain. Our test dataset includes 11,145 low star-rating user reviews collected from six apps following the inclusion/exclusion criteria described in Section 3.3. These apps are the mental health apps eMoods and Happify, the investing apps Wealthfront and Stockpile, and the food delivery app goPuff and Delivery.com. Table 3.4 describes the apps in our test dataset.

Before generating the summaries, for each of the reviews in our test dataset, English stop-words (e.g., *the, in, will*) are removed based on the list of stop-words provided in NLTK [66]. The remaining words are then lemmatized. We also exclude reviews that contain less than five words. This step is necessary in order to capture more informative summaries [127]. Table 3.5 shows the top summary reviews generated using Hybrid TF.IDF as well as our seeded summarization algorithm for each domain. For page-limit considerations, we only show the top five reviews. The table also shows the score calculated for each review and the concern category raised in the review, if any [150].

The results show that the majority of summaries generated by Hybrid TF.IDF contains valid user concerns. However, none of these concerns are privacy-related in any of the domains. Instead, the top five spots in all three summaries are hijacked by the most

dominant concerns in each domain, such as the app being inaccessible due to high fees or being unavailable to conduct a transaction. We further notice that there is a high level of redundancy in these summaries, even though a relatively low cosine similarity threshold of 0.2 was used. For instance, the summaries generated for the mental health, investing, and food delivery domains mainly expressed concerns regarding the apps being too expensive, untrustworthy, or having bad customer service respectively.

A deeper look into the data reveals that the generated Hybrid TF.IDF summary reviews contain the most important words in the corpus (based on their Hybrid TF.IDF score). Table 3.6 shows the top 10 words with the highest Hybrid TF.IDF scores in each application domain. These words appear in the summary reviews in each domain. For example, the summary reviews for the food delivery domain contain the words *"delivery", "time"* and *"service"*, which are the most important in this domain according to their Hybrid TF.IDF. In general, none of the domain-specific privacy-related keywords (see Table 3.2) are among the top 100 words in our corpora of user reviews. In fact, according to Hybrid TF.IDF, the keyword *privacy* is ranked 863, 2045, and 2004 in the mental health, investing, and food delivery domains, respectively.

Table 3.5 also shows that our proposed seeded summarization approach managed to overcome Hybrid TF.IDF's limitations in all application domains. By adjusting the importance of the privacy-related keywords, we raised the probability of the privacy-related reviews being included in the summary. The table shows that precision of 80%-100% can be achieved in all domains at 5-review length summaries. We further notice that our generated summary reviews experience less redundancy than the summaries generated by Hybrid TF.IDF. For example, in the mental health domain, four out of the top five reviews

raise privacy concerns about apps collecting personal information, demanding access to social media, and sharing user information with third-party entities. In the food delivery domain, the summary reviews raise concerns about apps collecting personal information, demanding access to phone contacts, and selling user information to third parties. In general, the low redundancy (higher coverage) in the summaries can be attributed to the fact that word embeddings are used to calculate the pairwise semantic similarity between reviews rather than relying on the textual similarity of their words. This helps to overcome the vocabulary mismatch problem affecting Hybrid TF.IDF. For example, the review *"I don't care how vetted this app is, no way are you getting my social and bank credentials"* is excluded from the summary of investing apps due to having high GloVe similarity with the summary review *"I'm worried because it has my SSN and bank login"*. Our analysis shows that a GloVe similarity score in the range [0.4 - 0.6] can achieve a balance between minimizing the redundancy of generated summaries and excluding important concerns.

To evaluate the performance of our seeded summarization approach at different length summaries, we generate summaries of lengths 5, 10, and 15 reviews for each set of reviews sampled from each of our domains. We assess the performance using two measures, precision and redundancy. Precision is calculated as the percentage of reviews in the summary that are privacy related, while redundancy is the percentage of reviews that raise privacy concerns already raised in the summary. In an ideal scenario, the precision should always hover around 100% while redundancy should be kept to the minimum (depending on the number of privacy concerns in the reviews). However, in reality, with more reviews included in the summary, we should see a drop in the precision and an increase in the redundancy, but with an increase in the recall, or the number of privacy concerns recovered.

Table 3.5: The top five reviews generated by Hybrid TF.IDF and our seeded summarization algorithm for all domains along with the average Hybrid.TF.IDF scores of each review and the quality concern expressed in the review.

Domain	Method	Top five reviews	Score	Concern
Mental Health	Hybrid TF.IDF	*"The app crashes every time I try to open it. This is not making me happy."*	0.013	reliability
		"Absolutely not worth it unless you want to pay almost 300$ a year, don't waste your time."	0.012	accessibility
		"Make it a one time purchase, not a subscription.would get 5 stars if I could have paid 4.99 for the app once not every month."	0.013	accessibility
		"The free activities are childishly simple and have nothing to do with actually creating a good mood, or anything other than taking up your time."	0.010	accessibility
		"Hasn't anyone ever heard the saying "money can't buy happiness"?"	0.008	accessibility
	Hybrid TF.IDF + GloVe	*"As soon as an app forces me to SIGN UP with Facebook (I gave up Facebook long ago and would never go back) or give my email... it's over, I am not giving out any personal info."*	0.17	privacy
		"One more paid app, that doesn't say clearly it is paid until collect your personal info"	0.17	privacy
		"Signups with email doesn't recognize my valid email as correct."	0.16	reliability
		"You shouldn't need access to any info on my phone that affects my security."	0.11	privacy
		"Patient, doctor confidentiality breeched. Your private chat is accessed by 3rd party without your permission."	0.009	privacy
Investing	Hybrid TF.IDF	*"If your looking to get your money taken this is a good app to start with."*	0.010	fraud
		"This app doesn't let you sell until the end of the day so by that time price can go way down do not use this app"	0.010	usability
		"Would have wanted to rate it higher but i cant even use the app because i cant even use my email saying that there already is an account."	0.009	reliability
		"Awful. High fees. Takes over 24 hours to trade every time"	0.008	usability
		"They locked my account. It says that it takes form 3 to 5 business days but it being 9"	0.008	usability
	Hybrid TF.IDF + GloVe	*"app is glitchy & does not connect with bank properly. now I'm worried because it has my SSN and bank login info. would not recommend. Google doesn't seem to vet any of these apps!"*	0.33	privacy
		"I can receive money from my bank but I cant send money to my bank"	0.29	usability
		"Like the app but it was constantly using my camera. There's no good reason for a stock trading app to use my camera especially when I'm not using it"	0.23	privacy
		"Let me link my bank account MANUALLY using routing/account numbers. I don't want to give you MY BANK ID AND PASSWORD. "	0.17	privacy
		"DONT USE THIS APP ITS A SCAM!! They SOLD my info and my social security number!!! "	0.17	privacy
Food Delivery	Hybrid TF.IDF	*" Ordered then got a call that they don't deliver to my area, after being told by the app that they did."*	0.016	accessibility
		"Cant get ahold of customer service, never got my delivery but was still charged. "	0.016	usability
		"Used to be great when it only took 20 to 30 minutes, but now every order takes one to two hours. "	0.014	usability
		" Makes you sign up before you find out they're not in your area. Waste of time"	0.012	accessibility
		"they made me wait one hour and i called and they told me i would ha e to wait another hour thats dumb hpnestly its happened twice already"	0.012	usability
	Hybrid TF. IDF + GloVe	*"not in my area, wish you asked for the zip first before i gave you all my info"*	0.29	privacy
		"Why do you need to download my contacts from email and my phone? Permissions are sketchy."	0.25	privacy
		" They sell screen recordings of customers to third partys"	0.17	privacy
		"This app wants too much personal information, unrelated to its purpose. Be aware. "	0.14	privacy
		"Ever time I go 2 update my acct info it crashes!! Has never worked"	0.14	reliability

Table 3.6: The top 10 words with the highest Hybrid TF.IDF in each of our application domains.

Domain	**Top 10 words**
Mental Health	*pay, free, get, use, try, subscription, money, want, time, trial*
Investing	*stock, market, money, buy, use, trade, get, make, fund, hedge*
Food Delivery	*service, get, delivery, time, food, use, customer, driver, bad, cancel*

The results of our analysis are shown in Fig. 3.3. A summary of size 10 seems to achieve a balance between precision and redundancy. At 10 reviews, we are able to maintain a relatively high precision while keeping the redundancy under control. Increasing the summary length from 5 to 10 in the mental health domain generates 5 more privacy-related reviews among which three are redundant. However, the review *"Really difficult to delete an account, a violation of privacy"* reveals a privacy concern (right to delete) that is not captured in the summary of size 5. Our analysis shows that setting the summary length to more than 10 can lead to a sizable drop in precision and a spike in redundancy.

In general, our results indicate that most domains have between 4 - 6 unique privacy concerns. To confirm this observation, we generate the top 50 summary reviews for each application domain. Following a systematic coding process similar to the process described in Sec. 3.4.1, we categorize the different types of privacy concerns present in these reviews. The main question to answer is: *what types of privacy concerns are raised in this review?* The results are shown in Table 3.7. Among the generated review summaries, around 28% are not privacy-related. In the mental health domain, the mandatory request for Facebook credentials or email addresses is the most dominant concern. In the investing domain, the dominant concern is related to apps requesting access to users' cameras and microphones. In the food delivery domain, 22% of generated review summaries are raising

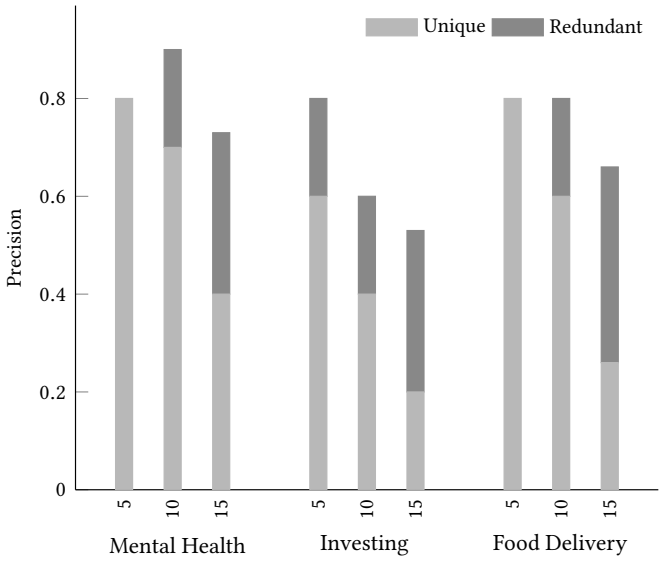


Figure 3.3: The performance of our summarization algorithm at different summary lengths as measured by precision and redundancy.

concerns about the unnecessary data collection practices of apps. In general, 4-5 unique privacy concerns are detected in each domain.

3.6. Discussion and Impact

Static code analysis of large datasets of apps in the mobile app market has revealed that the majority of apps do not comply with the privacy claims in their privacy policies [10, 151, 7]. Apps use intrusive data collection strategies to harvest more information than they need. Such information is frequently exploited to build addictive or habit-forming apps or even shared with third-party ad agencies [152]. Recent research efforts have been focused on analyzing the privacy practices of mobile apps. The objective is to inform mobile app users about the potential misuse of their personal data [7]. However, less effort has been made to keep app developers up-to-date with their end-users privacy concerns. This can be mainly attributed to the fact that such concerns are typically buried within more common types of usability, reliability, or accessibility concerns.

Table 3.7: The most common privacy concerns raised in the generated summaries (length of 50 reviews) for each of our subject application domains and their occurrences (Occ).

Dataset	**Concern**	**Occ.**
Mental Health	Requesting email/Facebook data	30%
	Asking for sensitive information before the free trial	20%
	Collecting patient information (health info, job history, lastname)	16%
	Requesting unnecessary permissions (location, device data)	12%
	Sharing and selling users' data	8%
	Other	12%
	Not privacy-related	24%
Investing	Requesting access to camera and microphone	26%
	Collecting identification information (SSN, birthdate, driver's license)	24%
	Collecting Financial information (bank statements, income, tax info, login info)	22%
	Other	10%
	Not privacy-related	30%
Food Delivery	Asking for personal information before trial use	22%
	Requesting unnecessary permissions (microphone, camera, contacts)	20%
	Asking for users' credit card information	10%
	Selling and sharing personal information to third parties	10%
	Other	8%
	Not privacy-related	32%

In this chapter, we propose a novel approach for extracting privacy concerns from user reviews. Our analysis is conducted using a dataset of app reviews collected from multiple application domains. Our first observation is that, due to their sparsity, privacy concerns can be easily missed. Our results also show that relying on generic keywords, such as *privacy* or *security*, often leads to omission problems, where the majority of domain-specific concerns are not identified. Therefore, domain knowledge should be used to guide automated algorithms toward privacy-specific issues. Our work in this chapter shows that privacy indicator keywords in a domain can provide such knowledge. The idea of using keywords has long been used to address the label scarcity bottleneck [153, 141]. Seeds that are derived from domain knowledge can guide text classification and topic modeling algorithms towards critical, but under-represented themes in the data.

In terms of expected manual effort, an argument could be made about the first phase of our analysis, where privacy-indicator keywords have to be extracted manually. However, our results show that using only statistically representative samples of reviews is enough to extract the most important keywords, where most keywords should be detected after only a few rounds of manual inspection. Therefore, the expected manual effort can be less significant than manually labeling an entire dataset of reviews for supervised classification tasks.

Based on our findings, we can conclude with a large degree of confidence that leveraging domain knowledge in text summarization can help to produce more concise and more descriptive summaries of privacy concerns in mobile app reviews. Summarization techniques have the advantage of being unsupervised; no large datasets of ground-truth data need to be labeled based on pre-defined labels (e.g., SUR-Miner [154] and MARS [122]). Generated summaries can also be more easily interpretable than the results often generated by standard text classification methods, where classified reviews need to be individually vetted to extract the most common privacy complaints present in the reviews. Instead, summarization algorithms only produce a small number of representative reviews that encompass the main themes in the reviews.

Summarization techniques can also have an advantage over unsupervised topic modeling techniques, such as LDA. To support this claim, we examine the performance of LDA in capturing privacy concerns in our test dataset of reviews [33, 27, 155]. We first apply text processing to enhance the quality of generated topics. In particular, we remove non-ASCII characters and exclude English stop-words. Lemmatization is applied to the resulting list of words. LDA hyper-parameter α is set to be automatically learned from the

corpus and β is set to 1/(number of topics). To ensure the stability of generated topics, the number of iterations (burnout) for the sampling process is set to 1000 [77]. To determine the number of topics, we rely on Gensim's coherence score. Topic coherence provides an objective measure to judge how good a given topic model is. Our analysis shows that, at around 5-8 topics, LDA generates the most cohesive topics for our dataset.

The most probable five topics generated by LDA for each domain are shown in Table 3.8; none of the generated topics seems to encompass any coherent theme related to privacy. For instance, the third topic generated for the investing apps includes privacy-related keywords such as *"card"* and *"info"*. However, the topic also includes other irrelevant terms such as *"stock"* and *"time"*, which commonly appear in reviews related to the availability of the service at a specific time frame. In general, most generated topics point toward issues that have been detected by classical Hybrid TF.IDF, including issues related to apps being expensive, such as the topic {*get, time, work, free, try, make, pay, money, version, one*} generated for the set of investing apps, or bad customer service, such as the topic {*service, order, customer, available, never, area, deliver, time, restaurant, item*} generated for the set of food delivery apps. These results can be largely attributed to the sparsity of privacy reviews (topics) in the data and their limited length [111, 78, 87]. Prior evidence has shown that LDA does not perform well when the input documents are short in length [111, 78, 87]. This leads LDA to downgrade topics related to privacy in favor of more prevalent topics, such as usability or reliability.

In terms of impact, our work in this chapter bridges an important gap in mining mobile app users' privacy concerns. Our expectation is that such information can help app developers to adjust their release engineering strategies to mitigate their end users' privacy

Table 3.8: The top five topics generated by LDA for the app reviews in our test dataset.

Dataset	Topics	Most probable words
Mental Health	Topic 1	get, time, work, free, try, make, pay, money, version, one
	Topic 2	time, use, pay, free, even, like, mood, need, activity, month
	Topic 3	pay, game, money, day, log, happiness, option, email, account, version
	Topic 4	like, time, better, work, get, make, way, pay, subscription, feel
	Topic 5	work, people, like, money, time, free, premium, help, great, happy
Investing	Topic 1	account, money, stock, even, fee, get, take, time, day, bank
	Topic 2	stock, money, day, account, sell, bank, customer, market, trade
	Topic 3	account, stock, time, money, buy, bank, card, log, back, sell
	Topic 4	stock, trade, time, price, money, buy, fee, make, use, email
	Topic 5	account, bank, get, service, email, money, customer, stock, use, still
Food Delivery	Topic 1	use, order, work, time, even, get, like, service, address, screen
	Topic 2	area, deliver, order, get, time, say, number, still, phone, email
	Topic 3	service, order, customer, available, never, area, deliver, time
	Topic 4	order, time, hour, deliver, never, minute, get, driver, service, food
	Topic 5	deliver, order, price, item, get, like, time, driver, customer, food

concerns and sustain their trustworthiness [119, 120, 121, 142]. This can be very critical for domains such as public health, where apps typically demand access to more personal information than the average app. For instance, health departments across the world have been using virus-tracking mobile apps to track down Covid-19 outbreaks. However, recent surveys have shown that a large percentage of the world population abstained from installing these apps due to privacy and mistrust concerns [156, 157]. Addressing these concerns can enhance these apps' adoption rates, thus contributing to the world's ongoing effort to fight the Covid-19 pandemic [158].

3.7. Threats to Validity

The study presented in this chapter has several limitations that could potentially limit the validity of the results. The main threat to the external validity of our study stems from the fact that only the top apps from three application domains were considered in our analysis. This could limit the generalizability of our results over other apps,

domains, or even less popular apps from these domains. However, given their large user-bases, popular apps often receive significantly more feedback than less-popular apps [159]. Therefore, privacy issues are more likely to manifest over these apps than smaller ones. Furthermore, we evaluated our approach over an unseen-before test dataset of apps. This has helped to enhance the confidence in the external validity of our approach.

A potential threat to the internal validity of our study might originate from the fact that, in the first phase of our analysis, domain experts were used to manually label privacy-related reviews and privacy indicative keywords. To enhance the validity of this process, a discussion session was held before running the labeling sessions to make sure that all experts were clear on their assignments and that all of their questions and concerns were addressed. These sessions included labeling samples of reviews to test-run our procedure. Furthermore, each expert only had to examine a statistically representative sample of reviews if the number of retrieved reviews was more than 300. No time constraint was enforced to minimize fatigue. Overall, these measures helped to preserve the integrity of the manually-labeled data; a small conflict rate of $\sim 5\%$ was detected between domain expert classifications.

Other internal validity threats might arise from the app review sampling process. In particular, we only included low-rating reviews in our analysis. This might have led to the exclusion of some informative reviews from the data. However, as recent evidence has shown, reviews expressing user concerns, and especially privacy concerns, are often associated with low star-ratings [16, 160, 141]. Therefore, excluding high rating-reviews is highly unlikely to lead to concern omission.

3.8. Conclusion

In this chapter, we proposed a novel unsupervised summarization approach for detecting privacy concerns in mobile app reviews. In the first phase of our analysis, we used an iterative word generation process to extract keywords indicative of privacy issues in three different mobile app domains. Our analysis showed that users in different application domains use different vocabulary to express their privacy concerns. This vocabulary is mainly related to the features of the app and its operational characteristics. In the second phase of our analysis, extracted keywords were used as seeds to help Hybrid TF.IDF, a generic text summarization technique, extract privacy-related reviews. Our evaluation showed that seeding Hybrid TF.IDF with domain-specific keywords helped to generate privacy-focused summaries. The results also showed that using word embeddings to calculate the semantic similarity between extracted summary reviews reduced the redundancy of generated summaries for each application domain. Our proposed approach is intended to help mobile app developers working in agile teams to quickly and accurately identify the most pressing privacy issues in their domain of operation, and ultimately, propose design solutions to mitigate these issues and enhance their chances of survival.

The work presented in this chapter will be extended along three main directions. First, we will continue to evaluate the proposed approach against other existing approaches and over other application domains. Our objective is to generate catalogs of comprehensive taxonomies and even NLP patterns that are indicative of privacy issues in the mobile app market [161]. Second, the generated taxonomies will be used to systematically tune different text summarization, modeling, and classification techniques and

identify near optimal configurations (e.g., summary length, similarity thresholds, etc.) to calibrate these techniques. Third, working prototypes will be developed to assess, through longitudinal studies, the usability and long-term practical significance of our approach. Ultimately, our objective is to help app developers understand how their end-users perceive their app's privacy practices and how these practices can impact their ratings and retention rates [142].

Chapter 4. Privacy Labels

Mobile applications (apps) are expected to disclose their data collection practices in their privacy policies. However, in reality, these policies are often verbose and difficult to understand, leaving sensitive user data vulnerable to privacy violations. To address these concerns, mobile app stores have recently rolled out a new form of privacy labels. These labels are intended to help app users understand the privacy practices of their mobile apps using simplified notations. In this chapter, we qualitatively analyze such labels across multiple application domains. Our analysis is conducted using a dataset of 90 apps, sampled from the domains of ride-hailing, mental health, and investing. In the first phase of our analysis, we explore the information value of privacy labels in the Apple App Store. We further measure the discrepancies between apps' privacy policies and their declared data labels. Our analysis shows that privacy labels can be used to compare the data collection practices among different application domains and detect privacy outliers in individual domains. Our analysis also exposes several types of inconsistencies between apps' privacy labels and their data collection claims in their privacy policies. Based on these findings, we propose several design strategies to help app stores enhance the utility and effectiveness of their privacy label systems.

4.1. Introduction

Mobile applications (apps) have become an integral part of our daily lives, changing the way we communicate, socialize, manage our finances, exchange information, eat, sleep, and exercise. However, in order for them to operate, apps constantly demand access to sensitive user information, including their geo-locations, usage patterns, financial

information, and health status data. Mobile app stores require all their hosted apps to provide privacy policies in which their data practices are described and justified [11]. Privacy policies are expected to provide information about the types of data apps collect, and how such data is being used, shared, transferred, and protected [151, 56]. However, in reality, such policies are often verbose, ambiguous, and jargon-heavy, making them difficult to understand by the average app user [7, 162]. To overcome these limitations, app stores have recently rolled out a new form of privacy labels. Similar to food nutrition labels, app developers can use privacy labels to declare the types of sensitive user data their apps collect. For example, the Apple App Store lists 14 different data types that apps may collect, such as location, search history, and contact information (e.g., Fig. 4.1). App developers have to declare the exact categories of data their apps collect before publishing their apps on the App Store, thus help their end-users get a sense of their privacy practices before downloading the app. According to Apple, *"users can learn about some of the data types the app may collect, and whether that data is linked to them or used to track them."*

In this chapter, we present an in-depth analysis of privacy labels in the Apple App Store. Our analysis is conducted using a dataset of 90 apps, sampled from the domains of mental health, ride-hailing, and investing apps. Apps in these data-driven domains rely heavily in their operation on users' willingness to share their private information with their apps. In our analysis, we explore how such apps express their data collection practices using privacy labels, investigate how these practices differ across application domains and within the same domain, study apps that are considered privacy outliers in their domains, and expose different types of inconsistencies between apps' privacy policies and their declared privacy labels.

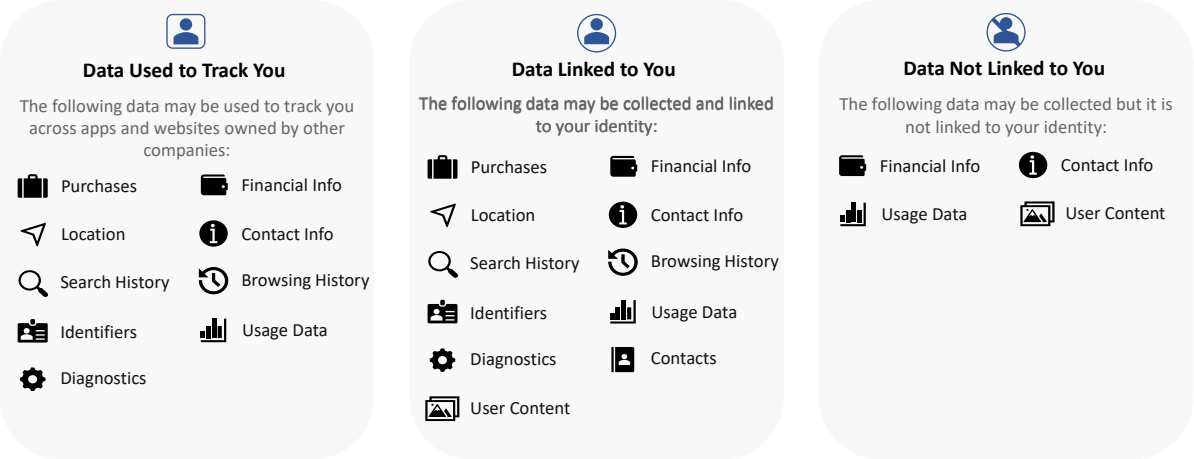


Figure 4.1: Grab app's privacy labels on the Apple Apple Store.

Based on our analysis, we suggest several design strategies to help app stores preserve the credibility and utility of their label systems. Our suggestions take the form of visual nudges that are designed to steer app users towards apps that preserve their privacy, and at the same time, prompt app developers to keep their data collection practices in-check. Recent evidence has revealed that apps that do not explicitly address their end-users' privacy concerns are often abandoned, or even deleted, by users [163, 156, 157, 164].

The remainder of this chapter is organized as follows. Section 4.2 motivates our research and describes our contributions. Section 4.3 describes our subject application domains and our experimental dataset. Section 4.4 presents our qualitative analysis of the privacy labels of apps in our dataset. In Section 4.5, we analyze our apps' privacy policies and measure their alignment with their privacy labels. Section 4.6 discusses our key findings and their potential practical implications on app store design. Section 4.7 addresses the main limitations of our study. Finally, in Section 4.8, we conclude the chapter and describe our future work.

4.2. Background

The concept of privacy labels was first introduced in 2009 by Kelley et al. [165, 166]. The authors suggested using some sort of nutrition labels to help websites better communicate their privacy policies with their end-users using visually appealing formats, or labels. In 2020, the Apple App store has rolled out its privacy labels (a total of 14 data types), demanding all app developers to self-declare their data consumption practices to their users. For example, Fig. 4.1 shows the privacy labels for the app Grab, a mobile app which provides its users with transportation, food delivery, and digital payment services. The labels, as they appear on the Apple App Store, are categorized into three main categories: Data Used to Track you, Data Linked to You, and Data Not Linked to You. These categories are intended to bring users' attention to the most sensitive types of information the app collects and how they are being used, or potentially abused.

While the research on mobile app privacy policies has made significant progress in recent years [10, 7, 162], the research on privacy labels is still in its early stages. Li et al. [167] conducted a series of interviews with 12 iOS app developers to identify the challenges they face when filling out the privacy labels of their apps. The results revealed that app developers, in general, viewed privacy labels as being helpful to users and developers. However, filling out these labels was perceived to be a challenging task due to label ambiguity and misunderstanding of Apple's definition of some of these labels, especially for apps that collect various types of user data. In terms of errors, the authors found common occurrences of false negatives (under-reported data practices) and false positives (reporting more data types than they actually collect). In a follow-up study, Li et al. [168] ex-

amined how well iOS developers complied with the new privacy requirement of the Apple App Store. The authors collected weekly snapshots of privacy labels of 1.4 million apps from April 2021 to November 2021. A qualitative analysis of the data showed that 51.6% of apps did not fill out their privacy labels and 5.8% of apps never updated their privacy labels. In another related study, Xiao et al. [169] presented Lalaine, a methodology to evaluate the data-flow to privacy-label consistency. The authors used dynamic analysis to identify inconsistencies between the actual data practices and privacy labels on a sample of 6,332 apps. The results showed that 64% of analyzed apps collected several data types without any disclosure in their privacy policies or labels, and 45% of apps did not fully disclose the full purpose for collecting one or more of their data types.

Building upon this line of research, in this chapter, we conduct a *first-of-its-kind* domain-specific analysis of privacy labels in the App Store. In particular, we measure the extent to which such labels can be used to understand and compare the privacy practices of apps across multiple domains and explore ways for improving their credibility and utility. Our main motivation is to propose design strategies to help app stores *nudge* their users to make more privacy-aware app selection decisions. Digital Nudging theory posits that it is possible to architect a digital choice environment in a way that individuals can enjoy the freedom of choice but are *"nudged"* towards choices that are more beneficial for them or society in general [170]. The choice architecture is the context in which people make decisions, namely a manipulated choice environment [171]. Drawing on the main principles of digital nudging, our analysis aims to architect a choice environment (app privacy label system) that is optimized to direct users towards apps that preserve their privacy. To achieve this goal, in this chapter:

- We investigate the types of information that can be learned about apps from their privacy labels, including how these labels differ among application domains and within the same domain. Our goal is to help mobile app users identify privacy outliers in their domain of interest as well as help app developers to learn the privacy practices of their domains, thus adjust privacy practices to avoid being an outlier.

- We compare the privacy claims of apps as declared in their privacy policies to their marked privacy labels. Our objective is to identify and classify the types of errors in data label reporting and help app developers align the data collection claims in their privacy policies with their privacy labels.

- We propose several design suggestions to help app stores enhance the overall utility and effectiveness of their privacy label systems.

4.3. Subject Domains and Dataset

The objective of our analysis is to study the privacy practices of mobile apps across and within different application domains through their privacy labels. To conduct our analysis, we curated a dataset of 90 mobile apps from three application domains in which apps commonly operate on sensitive user information. In what follows, we describe the privacy practices of apps in these domains.

4.3.1. Ride-Hailing

Ride-hailing apps, such as Uber and Lyft, have transformed the way we travel by providing a convenient on-demand model for connecting drivers and riders [172, 173]. Similar to other types of sharing (*gig*) economy apps, privacy in the ride-hailing domain is a

compounded concept, including interrelated concerns of data and physical privacy [174]. Fig. 4.2 shows the different channels of data exchange in a typical ride-hailing transaction [175]. To be approved as a service provider (e.g., Uber driver), users need to disclose their personally identifiable information (PII) to ride-hailing companies (e.g., Uber or Lyft), including their legal names, addresses, and pictures along with information about their personal assets, such as their vehicle information and their preferred working hours. Service consumers (e.g., Uber riders) also need to share their PII and financial information with ride-hailing companies in order for their service requests to be approved. Ride-hailing companies use this information to mediate their transactions and charge for the service. Some of this information is also made available to service providers and receivers to facilitate identification offline.

Recent analysis of the privacy practices of ride-hailing apps has revealed that these apps often operate under vague privacy policies [176, 174]. Major concerns are also frequently raised about their invasive geo-location tracking along with concerns about inferring riders' routine habits and consumer preferences [176]. Recent evidence has further revealed that such concerns can translate into attacks on physical privacy, including driver/rider harassment, stalking, and abduction [177, 174].

4.3.2. Mental Health

Mental health apps have gained considerable popularity among patients and doctors over the past decade. Such apps generate behavioral interventions to help their users deal with a variety of mental illnesses, including depression, anxiety, addiction, and eating disorders. In fact, the availability of such apps has been found to reduce barriers to access-

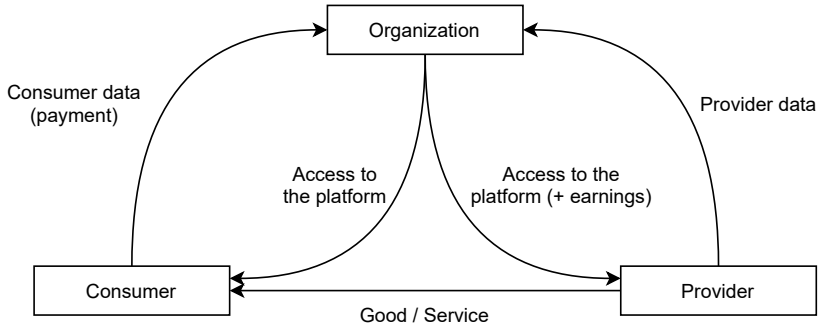


Figure 4.2: A summary of data exchange between users and companies of sharing economy apps [175].

ing mental health services, including the high cost and negative stigma commonly associated with receiving treatment for mental illness [178]. Over the past two years, the global health crisis that was caused by the Covid-19 pandemic has led to a significant spike in the number of active users of mental health apps. People frequently resorted to these apps as a safer and inexpensive alternative to help them cope with the mental consequences of social isolation, unemployment, and economic hardships [137, 138].

Despite their benefits, major concerns are often raised about the transparency of mental health apps when it comes to their data collection practices [50]. In particular, to deliver tailored health interventions, these apps need to collect sensitive medical information from their users about the status of their mental health. This poses a substantial risk to users' privacy as leaking this type of information can have serious repercussions on their personal and professional lives. However, recent studies have revealed that the majority of mental health apps either do not take necessary measures to preserve their users' privacy or share their data with undeclared third-parties [179]. In addition, their privacy policies are often too difficult to understand by the average user [180].

4.3.3. Investing apps

Investing apps have become increasingly popular in recent years due to the increasing interest in cryptocurrency trading. Zero-commission trading fees and continuous media coverage have brought in millions of new first-time traders. For example, one of the most popular investing apps, *Robinhood*, reported that close to six million new users joined the platform in 2021. In order for them to mediate financial transactions, investing apps demand access to large amounts of investors' data, including their Social Security Numbers (SSN), credit card information, and bank account information. While this information is necessary to prevent fraud and verify the legitimacy of online trades [181], the leak of such data can expose users to substantial financial and personal risks [182].

4.3.4. Experimental dataset

Our data collection took place in June of 2022. We identified the top-100 apps in the categories of Finance (Investing), Travel (Ride-hailing), and Health&Fittness (Mental Health) on the Apple App Store. Only apps that reported collecting data types in their privacy labels were included. In addition, for an app to be included in our analysis, we only considered apps with 1,000 reviews or more. This number of reviews was used as a signal that the app has a sizable userbases. Furthermore, apps without a public privacy policy were excluded. For the investing domain, banking *"all-in-one"* apps were excluded as the majority of these apps did not provide investing services. In the Health&Fitness category, physical health apps that did not explicitly support mental health were excluded. After examining and ranking the top 100 apps in each category, we selected the top 30 apps in each application domain. Table 4.1 shows these apps along with their popularity,

Table 4.1: A description of our dataset.

Domain	#**Apps**	Avg. **Rating**	**Reviews**[Min, Average, Max]	**Installs**[Min, Average, Max]
Ride-hailing	30	4.4	[1K, 424K, 9M]	[1.8K, 60M, 751M]
Investing	30	4.3	[1.2K, 301K, 4M]	[19K, 3.4M, 18M]
Mental Health	30	4.7	[1.1K, 104K, 1.3M]	[0.7K, 4.4M, 48M]

measured as the number of reviews they received, their average rating on the Apple App Store, and the number of installs from Google Play.

4.4. Analysis

In this section, we analyze the privacy labels of apps in our dataset at inter and intra application domain levels. Our objective is to explore how these labels are used by different apps in different application domains.

4.4.1. Inter-Domain Analysis

Under this phase of our analysis, we explore how apps in different application domains utilize privacy labels. To conduct this analysis, we track the data types declared by each app in each of our subject domains along with the justification provided for collecting each type, if any. Fig. 4.3 shows the number of apps that collect each data type in the ride-hailing, investing, and mental health domains. The table also shows the significance of the dependency between the domain and the data type.

Towards the bottom of Fig. 4.3, we notice that there are several data types that are collected by almost all apps in all domains. These types include contact info, identifiers, diagnostics, and usage data. Furthermore, as expected, there are several data types that are domain specific. For example, Health&Fitness data is only collected by mental health apps (n = 15). These apps mainly operate on users' mental health information, thus collecting this type of data is justified. Financial data is mainly collected by invest-

Private Data	RideHailing	Investing	Mental Health	χ^2
Browsing History	1	3	3	1.23
Sensitive Info	4	3	3	0.22
Health & Fitness	0	0	15	**36*** **
Contacts	11	3	2	**11.09** **
Other Data	7	7	4	1.25
Search History	15	10	8	3.73
Financial Info	15	20	1	**26.94*** **
Purchases	9	15	16	3.87
Location	29	10	13	**28.51*** **
User Content	21	18	15	2.5
Contact Info	28	23	21	5.41
Identifiers	25	27	27	0.82
Diagnostics	26	29	24	3.93
Usage Data	24	28	28	3.6

Figure 4.3: The number of apps collecting each data type in our different application domains. The significance of the dependency between the data type and the domain is measured using Chi-square (χ^2), $^{*}p \leq 0.05$, $^{**}p \leq 0.01$, $^{***}p \leq 0.001$).

ing (n = 20) and ride-hailing (n = 15) apps. Investing apps frequently request access to users' financial information, including payment information, tax documents, bank statements, and income to provide a variety of investing-related services, such as money withdrawal, depositing, and tax returns. Half of the apps in the ride-hailing domain (n = 15) declare access to their users' financial information (i.e., payment information). In general, this practice is justified to process payments. In the mental health domain, only one app, Sanvello, declares a demand for user financial information. Sanvello claims that it collects users' payment information for functionality purposes. However, the privacy policy of this app does not justify why such information is needed.

Users contacts (address book) information is mostly collected by ride-hailing apps (n = 11). These apps request access to their users' contacts to "*help you refer friends*" or "*add an emergency contact*". Only two mental health and three investing apps demand access to users' contacts. For instance, the mental health app Simple Habit justifies accessing users' contacts to allow users "*to provide information about your friends through*

our referral services." Browsing history and sensitive user information are the least collected data types in all domains. Only three investing, three mental health, and one ride-hailing apps collect and link users browsing history to their identities for product personalization, analytics, and advertising purposes. The ride-hailing app Grab is the only app that collects browsing history information, but without providing any justification for why such data is actually collected. Sensitive information (e.g., racial or ethnic data, sexual orientation, and disability) is another data type that is accessed by only three or four apps in each domain. For instance, Bridj—a booking and fleet management ride-hailing app—is the only app providing justification for collecting such information. According to Bridj's privacy policy, the app collects information about users' mobility and accessibility needs "*to assist service providers to provide mobility and transportation solutions.*"

Location information is another popular data type that is collected by several apps in each domain. Out of the ride-hailing apps in our dataset, 25 apps access users' precise location to search for nearby rides and facilitate accurate pickup and drop-off. Only 10 apps in the investing domain ask for users' location. Four of these apps collect users' coarse (approximate) location and the other apps (n = 6) request users coarse and precise location. For instance, the investing app Betterment requests that users enable their location feature "*to provide you with cash back offers in your vicinity*". The mental health apps in our dataset do not demand access to users' precise location, only 13 apps access users coarse location for marketing purposes or for improving users overall experiences. However, no further information is provided on how user experience is exactly improved.

We use Chi-square (χ^2) to measure if the number of apps collecting each data type varies significantly among the different domains. Our null hypothesis H_0 is that there is no

difference in the number of apps that collect a specific data type between all domains. The alternative hypothesis H_1 is in favor of the dependency between the data type and the domain. Since we have two variables (the number of apps that collect a specific data type and the number of apps that do not) and three groups (domains), the degree of freedom in our test is set to 2 = (2 - 1) * (3 - 1). Given this degree of freedom and a confidence level of 0.001, 0.01, and 0.05, χ^2 critical values are set to 13.816, 9.210, and 5.991, respectively. Fig. 4.3 shows the results. For the data types Health&Fitness, Contacts, Financial Info, and Location, we can reject H_0 with a confidence level of at least 0.01.

In addition to the general data types, we also analyze the sub privacy types declared by each app in each domain. Fig. 4.4 shows the number of apps collecting each sub-type of users' private information in our different application domains. The figure shows that, in the ride-hailing domain, phone number is the most collected sub-type of contact information, while in the investing and mental health domains, email address and name are respectively the most collected sub-types of contact information. When it comes to location, the majority of ride-hailing apps (n = 26) demand access to precise user location. Only six apps in the investing domain require access precise location, and none of the mental health apps demand access to this sub-type. User Content is another data type that seems to vary at a sub-type level. This type includes the sub-types of user generated data, such as their text messages, photos or videos, audio data, voice or sound recordings, gaming data, and customer support data generated by the user during a customer support request. Our analysis shows that almost half of the ride-hailing (n = 14) and investing (n = 16) apps collect users photos and videos. However, only three mental health apps require this particular sub-types of user content.

		Number of of apps collecting each data type		
Data Type	**Sub-type**	Ride-hailing	Investing	Mental Health
Browsing History	Browsing History	1	3	3
Contact Info	Email Address	26	23	20
	Name	25	22	20
	Phone Number	27	19	2
	Physical Address	15	18	1
	Other	4	8	2
Contacts	Contacts	11	3	2
Diagnostics	Crash Data	27	28	23
	Performance Data	20	25	17
	Other	18	18	9
Financial Info	Payment info	16	16	1
	Other	1	17	0
	Credit info	2	0	0
Health & Fitness	Health	0	0	13
	Fitness	0	0	2
Identifiers	User ID	24	26	26
	Device ID	24	21	23
Location	Coarse	18	9	13
	Precise	26	6	0
Other Data	Other Data	7	7	4
Purchases	Purchase History	9	15	16
Search History	Search History	15	10	8
Sensitive Info	Sensitive Info	4	3	3
Usage Data	Product Interaction	23	29	27
	Advertising Data	7	10	8
	Other	7	10	4
User Content	Customer Support	16	15	9
	Photos or Videos	15	14	3
	Other	4	6	9
	Audio Data	6	3	2
	Emails or Text Messages	3	2	0
	Gameplay Content	0	0	1

Figure 4.4: The number of apps collecting each data type and sub-type in each of our subject application domains.

To get further insights into the difference between the different domains, we examine the number of data types that are tracked (may be used to track users), linked (may be linked to the user's identity), and not linked to users' in each domain. The results are shown in Fig. 4.5. We observe that the way some data types (highlighted in the table) are utilized varies significantly within the same domain. For example, in the ride-hailing domain, diagnostics are linked to users identity in 17 apps, while 12 other apps collect such data but do not link it to their users. Similar patterns can be observed in other domains.

	Ride-hailing			Investing			Mental Health		
	Tracked	Linked	Not Linked	Tracked	Linked	Not Linked	Tracked	Linked	Not Linked
Browsing History	1	1	0	1	3	0	0	2	0
Contact Info	5	26	4	3	22	2	3	20	2
Contacts	0	7	6	0	3	0	0	0	2
Diagnostics	3	17	12	0	16	17	1	14	13
Financial Info	3	14	3	0	18	2	0	1	0
Health & Fitness	0	0	0	0	0	0	0	13	3
Identifiers	15	22	5	10	24	5	5	25	6
Location	4	23	8	2	10	1	1	10	3
Other Data	1	6	1	1	3	4	0	3	1
Purchases	3	9	0	3	15	0	3	13	2
Search History	2	9	6	2	8	2	0	7	0
Sensitive Info	0	4	0	0	3	0	0	3	0
Usage Data	9	21	7	7	22	6	4	22	7
User Content	1	17	4	0	18	2	0	13	1
Sum	**47**	**176**	**56**	**29**	**165**	**41**	**17**	**146**	**40**

Figure 4.5: The number of apps tracking, linking, and not linking each data type in each of our subject application domains.

According to the App Store, diagnostic data includes crash and performance data. While collecting this specific type of data is expected, it is not clear either in their label justification or privacy policies why almost half of the apps in our dataset link diagnostic data to their end-users' identity.

> ***Summary*** - *While there are some data types that are commonly collected by almost all apps in all domains, each domain has its own defining data types that are related to its core functionality. Some data types are more linked to users' identity across all domains than others. Furthermore, apps in different domains and within the same domain can vary in the sub-types of private user information they collect.*

4.4.2. Intra-Domain Analysis

The goal of our intra (within) domain analysis is to expose apps that do not conform to the privacy standards of their domains. In other words, apps that collect less or more data types than the average app in the domain. We refer to those apps as privacy outliers. To conduct such analysis, we generate a distribution graph for the number of data types collected by each app in our dataset. The results are shown in Fig. 4.6. In gen-

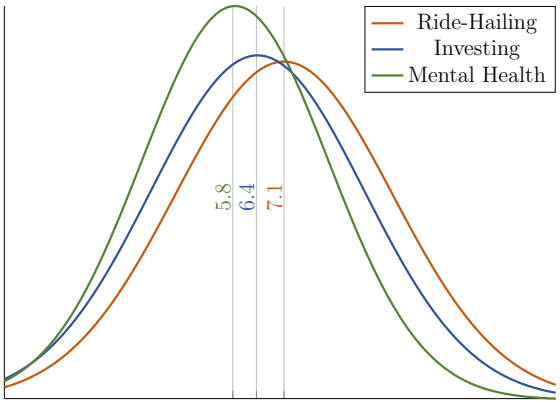


Figure 4.6: The distribution of the number of data types collected by apps in our subject application domains.

eral, apps within the same domain tend to follow a normal distribution when it comes to the number of data types they collect/declare. To simplify our analysis, we consider any app that collects three or less (≤ 3) or ten or more (≥ 10) data types in each domain to be an outlier. Table 4.2 shows the distribution of apps in each domain. Ride-hailing apps collect on average 7.1 data types. When considering the left side of the bell curve, we find that the apps Curb, Rapido, Veyo, and zTrip declare collecting only two types of information: Users' Location and Contact Information. On the other end of the curve, we find that six apps collect 10 or more types of user data.

In the investing domain, apps declare on average collecting 6.4 data types. A total of five apps declare collecting three or less data types. For instance, the investing apps Personal Capital and BitMart only collect diagnostics, users/devices identifiers, and usage data. However, in the same domain, four apps access more than 10 types of user private information. Stockpile and Robinhood are among these apps, declaring the collection of some data types (e.g., location and contacts) that are not commonly collected by other apps in the domain.

Table 4.2: Privacy outlier apps (collect more than 9 or less than 4 data types) in our dataset.

Domain	**Apps collected ≤ 3 data types**	**Apps collected > 3 and < 10 data types**	**Apps collected ≥ 10 data types**
Ride-hailing	Curb, Rapido, Veyo, zTrip	Beat, Bussi, Flywheel, Wingz, ADO, Empower, Moovit, Cabify, Revel, 99, BalBlaCar, Bolt, DiDi Rider, HopSkipDrive, Urbvan, Via, Waze, Gett, kakao, Scoop	Bridj NSW, MyBluebird, Ola, Grab, Uber, Lyft
Investing	Personal Capital, IBKR, BitMart, E*Trade, MarketSim	BlockFi, Public.com, Thinkorswim, Titan, Power E*trade, Rally Rd., Gemini, Wealthfront, Betterment, Invest&Tarde, Schwab, TD Ameritrade, Wealthsimple, Webull, Atom Finance, KuCoin, Stash, Vanguard, Fidelity Spire, Acrons, Fidelity Investments	Stockpile, Stocktwits, Uphold, Robinhood
Mental Health	What's up, MoodMission, Talkspace, BetterHelp, iBreathe, ReGain, Wysa	Smiling Mind, Aura, Mindfulness, Shine, Dare, Lifesum, Moodfit, Rootd, Slumber, Youper, Calm, Meditation Studio, Waking Up, Woebot, Zen, Happify, Happy Not Perfect, NOCD, Simple Habit Sleep, Headspace, Lumosity, Ten Percent Happier	Sanvello

Mental health apps collect on average 5.8 data types. Out of the 30 apps in this domain, six apps collect less than three types of user information. For instance, MoodMission and iBreathe collect only Users Diagnostics and Identifiers. However, Sanvello, another popular app in the same domain, collects more than 10 different types of private information. In fact, this particular app collects types of information that no other app in the mental health domain reports collecting, including Financial Information, Browsing History, and Sensitive Information.

Summary *- Apps with similar core functionalities can substantially vary in terms of data collection practices. Each domain seems to have a set of outlier apps that either collect more or less data types than the average app in the domain. Furthermore, apps within the same domain can significantly vary in terms of the data types they track or link to their users.*

4.5. Policy vs. Labels

In the previous section, our analysis of privacy labels and sub-labels has revealed that some apps do not seem to conform to the common privacy practices in their domains. To determine if these apps are in fact true outliers, in this section, we analyze their privacy policies. We start our analysis by extracting the data collection claims from the policies. Each claim is then mapped into one or more of the 14 data types (labels) and sub-types listed by Apple. In particular, each claim is broken down into the specific privacy label mentioned in the claim along with the justification—if any—for collecting that type of information. For example, in the privacy claim in Fig. 4.7, the clause *"we collect your location information"* is mapped to the App Store's privacy label *location*. The rest of the claim provides the justifications for collecting user location as provided by the app. In case of no suitable category or subcategory is found, the data claim is listed as *Other Data*. For example, the claim *"vehicle information and background check results"* frequently appearing in ride-hailing apps' privacy policies is mapped to *Other Data* as there is no specific label or sub-label defined for this type of information.

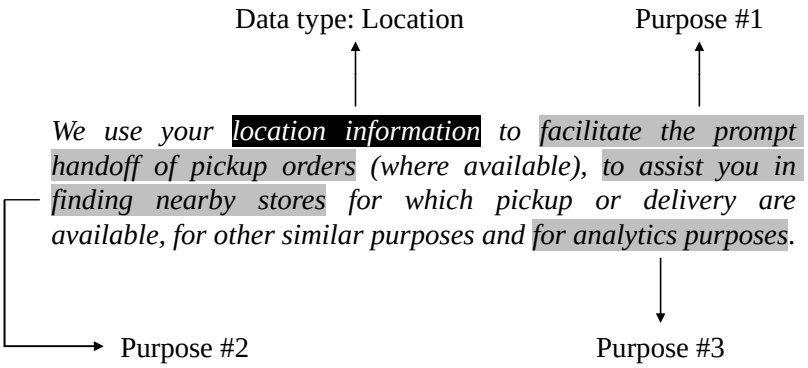


Figure 4.7: An example of mapping a data collection claim in a privacy policy to a specific App Store privacy label.

The analysis is conducted by two separate researchers to ensure the integrity of the data. A policy from each domain was first analyzed by both researchers to expose any misunderstandings or miss-communication about the specific task to be performed. The other policies were then analyzed individually. The results were then consolidated over multiple sessions where each individual policy was discussed. Conflicts and disagreements were resolved after further discussions and with the help of the App Store's formal description of their privacy labels. Conflicts were detected in less than 5% of claims. In general, most conflicts originated from vague privacy claims that were classified under *Other Data* (e.g., Table 4.3). Our multi-step classification process and the cross-checking of individual researchers classifications helped to maintain the integrity of the data. In general, our analysis has exposed significant inconsistencies between apps' privacy labels and their policies. These inconsistencies can be classified into two main categories:

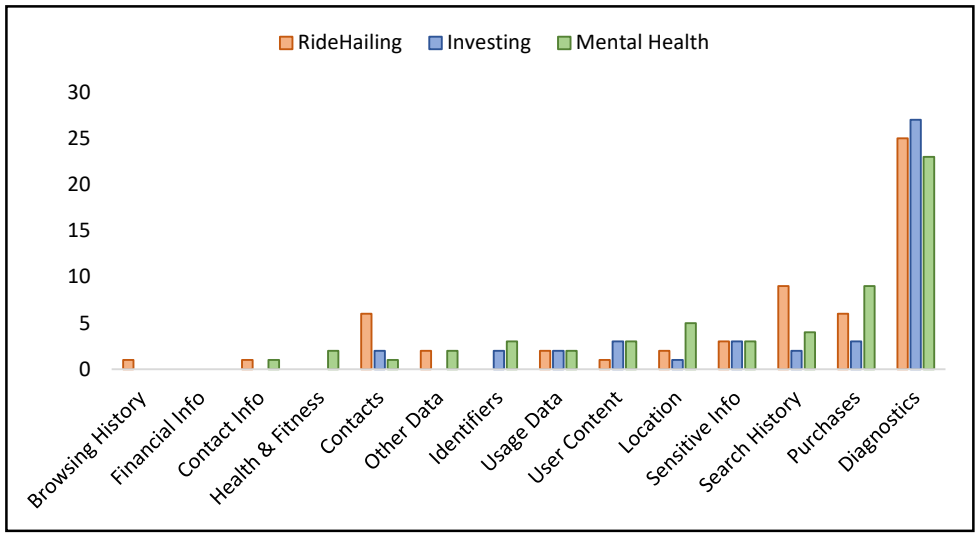
- Over reporting: this category of error refers to cases where some of the data types listed in the labels are not acknowledged in the policy. Fig. 4.8-a shows, for each data type, the number of apps that report collecting that data type in their labels but not in their privacy policies. The figure shows that the majority of apps in our application domains (n = 79) declare collecting diagnostics in their labels. However, most of these apps (n = 75) do not disclose collecting such information anywhere in their policies. Only four apps (one ride-hailing, two investing, and one mental health) declare collecting diagnostics in both of their privacy policies and data labels. For example, the investing app KuCoin reports that two third-party libraries are used to collect log data and crash information (i.e. diagnostics) for sta-

Table 4.3: Examples of data types that apps claimed collecting in their privacy policies and were mapped into "*Other Data*".

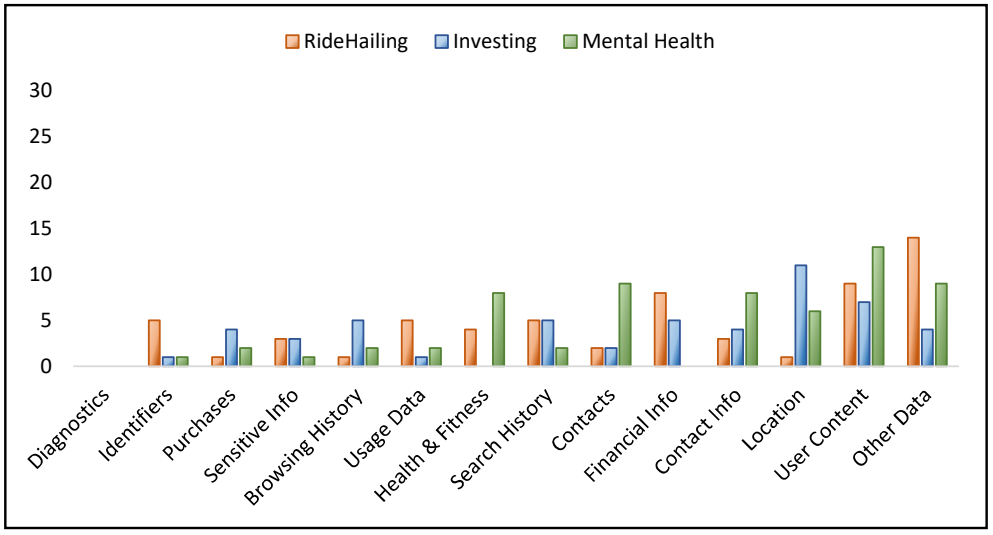
Domain	**Specific Data Type**	**Example Data**
Ride-hailing	Vehicle Information	details on your cars, insurance information, and vehicle registration card
	Telematics Data	speed, acceleration, and braking data
	Users Background Information	driving history, criminal record, and right to work
Investing	Sensitive Identifiers	Tax ID number, citizenship, passport number, visa information, Social Security Number, government-issued identification numbers, alien registration number
	Education/Employment History	Academic record, degrees and schooling, office location, job title, description of role, employer name, employment status
Mental Health	Social Media Data	Facebook profile information, such as name, email address, and Facebook ID
	Employment Information	employment history, education background, company name, company email address

bility monitoring purposes. Users' purchases and search history collection claims are also commonly omitted from privacy policies. For instance, 18 out of 40 apps that declare collecting purchase information in their privacy labels, do not report collecting such information in their policies. Similarly, 15 apps out of 33 apps that declare collecting users' search history in their labels do not include any data collection claims about this specific type of user information in their policies. On average, apps in the ride-hailing, mental health, and investing domains over-reported 1.9, 1.5, and 1.9 data types respectively.

- Under reporting: this category of error refers to cases where apps do not declare in their labels collecting specific data items that are reported in their privacy policies. Out of the 90 apps included in our analysis, 25 ride-hailing, 21 investing, and 25 mental health apps' policies include claims about data types that are not de-



(a)



(b)

Figure 4.8: Inconsistencies in reporting. a) over-reporting: the number of data types that are declared in the privacy labels but not in the privacy policies, and b) under-reporting: the number of data types that are claimed in the policies but are not declared in the privacy labels.

clared in their privacy labels. As Fig. 4.8-b shows, location, user content, and other data labels are the most commonly undeclared data types in the privacy labels. For instance, our results show that ride-hailing apps in our dataset under-report collecting user content, financial information, and other data. In the investing domain, location and user content are the most under-reported data types in the privacy labels, while user content and contacts are the two most under-reported data types by mental health apps. Fig. 4.9 shows the distribution of the number of unreported data types over our three domains. On average 2.2, 1.8, and 1.7 data types appear in the privacy policies of mental health, ride-hailing, and investing apps respectively but are not declared in the privacy labels of these apps.

Problems of under-reporting can be mainly attributed to the fact that data labels are not comprehensive. This is reflected in the fact that several data collection claims in the privacy policies of our apps do not map to any specific privacy label. Table 4.3 shows several examples of these data types along with their data collection claims in their apps' policies. As the table shows, some of these data types are too specific to the domain, such as telematics data (driver's speed and acceleration data) in ride-hailing apps, employment history in investing apps, or social media data in mental health apps. Under-reporting can be dangerous as it can deceive users into downloading an app based on the fact that it declares collecting less data types in its labels than other apps. However, given that the average number of under-reported types in each domain is relatively low for most apps ($\leq$ 2) and the fact that such problems often affect a small number of labels, we believe such errors can be corrected if a more comprehensive set of labels are used.

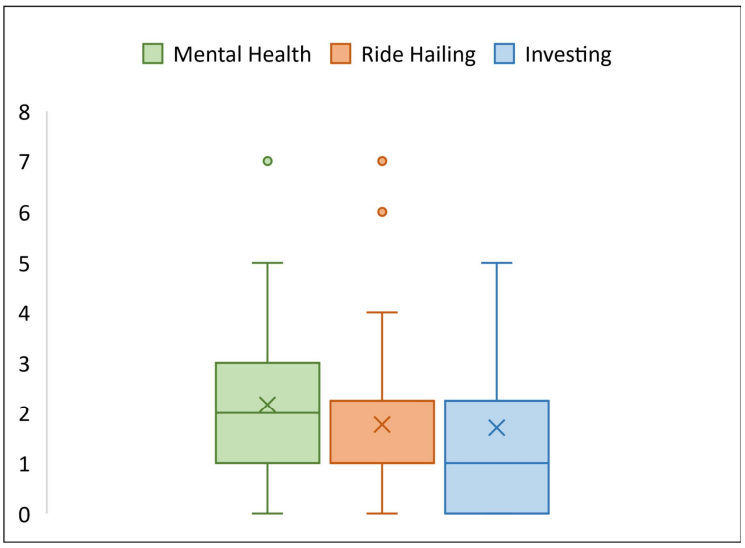


Figure 4.9: A boxplot of the number of under-reported data types in each of our application domains.

4.6. Implications

Recent evidence has revealed that privacy can substantially influence users' app selection decisions [158, 157]. Any form of privacy violation can severely undermine apps' trustworthiness, and ultimately, chances of survival in the app market. For example, health departments across the world have been using mobile apps to track down Covid-19 outbreaks. However, several emerging surveys have shown that a large percentage of the world population has abstained from installing these apps due to privacy and mistrust concerns, thus hindering the world's ongoing effort to recover from the Covid-19 pandemic [158, 156, 157].

Our work in this chapter looks at privacy labels in app stores as a strategic asset that can be utilized by app developers to enhance their trustworthiness, and thus, maintain healthy retention rates. At the same time, such labels can nudge users to make more effective app selection decisions. As mentioned earlier, the idea of privacy labels originated

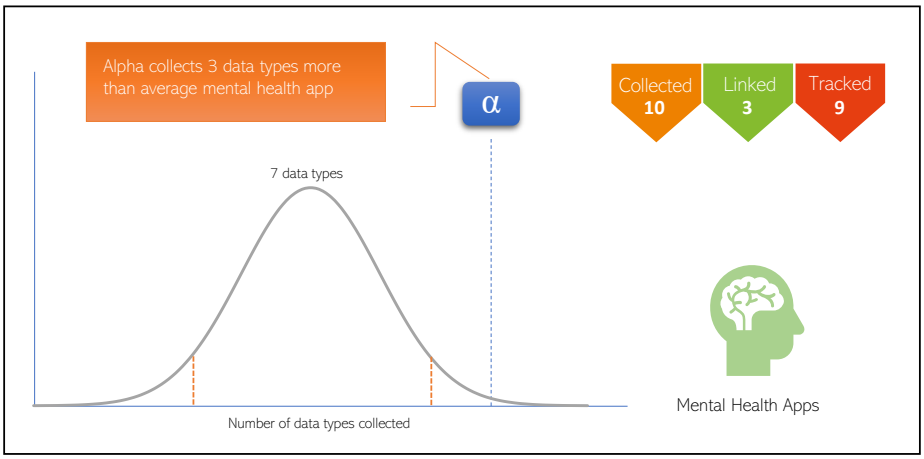


Figure 4.10: Suggested visual aid to show where an app (The mental health app Alpha) stands in terms of data practices in comparison to other apps in its domain.

from nutrition labels. Evidence from the food industry suggests that nutrition labels can be an effective source of nudges, significantly promoting healthy food consumption habits by influencing consumers' food purchase decisions [183, 184, 185, 186]. Following this evidence, and based on our findings, we make several recommendations to help app stores increase the utility of their privacy label systems. To ensure an effective delivery of nudges, our recommendations are designed to increase the salience of recommended choices (privacy preserving apps). Salience is one of the principal techniques behind the design of nudges [170]. We suggest to increase salience by novel, relative, and vivid interfaces that direct users' attention to particular choice options but at the same time maintain their liberty of choice [187]. These recommendations can be described as follows:

- ***Your app vs. other apps:*** we suggest displaying some sort of visual aid to represent where an app stands in terms of its privacy practices in comparison to other apps in its specific application domain (e.g., Fig. 4.10). This kind of information can help users make more informed app selection decisions by only selecting apps

that either conform to the privacy standards of their domain or apps towards the left side of the curve. Such a visualization will also prompt app developers (especially toward the right side of the curve) to readjust their data collection strategies to conform to the domain's standards (move to the middle part of the curve). However, as our analysis has shown, privacy is a domain-specific concept. Privacy practices are drastically different between application domains; what might be considered a privacy violation, or unnecessary data collection, in one domain, may be an accepted practice in another domain. Therefore, for our suggested nudge to work, it must show how an app performs relative to other apps in its specific domain. Existing app stores' categories include thousands of loosely related apps. In other words, they are too generic to be considered coherent application domains [28, 188], thus, app stores need to adopt a more fine-grained form of categorization for their apps to enhance the information value of their label systems. This can be achieved through app classification techniques that can automatically identify the operational domain of an app by analyzing the description of its features and identifying its domain of functionally-related apps [188, 27].

- ***Privacy Score*:** privacy is a compound concept; it is not enough to show how many data types an app collects. It is also important to show how many of these types are tracked or linked to the user's identity. However, as our analysis revealed, these numbers heavily depend on the application domain. Therefore, it makes sense to show where an app stands in terms of data collection, linking, and tracking relative to other apps in the same domain. In the food industry, color coding is used

to show such information. For example, Countries in the European Union (EU) have embraced the use of a color-coded rating system known as the Nutri-Score (Shown in Fig. 4.11) to show how healthy a food product is. Nutri-Score is designed to be an easy and non-intrusive form of visual aid that consumers can use to compare food products at a glance. The score itself is based on the calculation of an EU nutrient profiling system. Converging evidence suggests that this system has had significant positive impacts on promoting healthy eating habits and curbing obesity [189, 190]. Following this line of evidence, in addition to our previous suggestion, we suggest an app privacy labeling system similar to the Nutri-Score. The color-coding system (shown in the right part of Fig. 4.10) shows the number of data types the app collect, track, and link and how these numbers compare (through color coding) to other apps in the domain. Alternatively, a holistic domain-specific privacy-score which consolidates all these numbers into one number can be used to provide a *mental shortcut*, or reduce the cognitive effort, associated with deciding which apps to download [191].

- ***Your domain is changing:*** we suggest that app stores should periodically notify app developers about changes in the privacy practices of their domain. If an app has become an outlier that collects more data labels than its peers, app developers can take action to curb their data collection engine to remove some of the labels that make them stand out. Developers can also be notified if their app suspiciously collects less data types than other apps in its domain. The goal is to check whether such apps are true outliers or simply under-reporting their data claims.

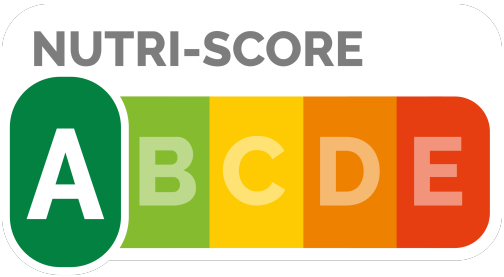


Figure 4.11: The Nutri-Score food labeling system used in the EU.

- ***Your privacy policy says otherwise:*** to enhance and maintain the integrity of the label systems, app stores can provide some sort of tool support to automatically check the alignment between apps' privacy policies and their declared data labels. This type of support can help developers detect errors of under and over-reporting. Such tools can utilize text classification and modeling techniques to map data collection claims in the policy to data labels. A more advanced implementation could employ static or dynamic analysis of apps code to detect cases of under and over reporting [6, 192].

We anticipate that, if implemented, the suggested nudges have the potential to substantially enhance the utility of the privacy label systems of app stores. Not only end-users will be able to select apps that preserve their privacy, but also app developers can learn more about their domain of operation, correct errors in reporting, and enhance their apps' trustworthiness and chances of survival.

4.7. Threats to Validity

The study conducted in this chapter has multiple limitations that could potentially limit the validity of our findings. The main threat to the external validity of our study stems from the fact that we only analyzed the top 30 apps in three application domains.

Analyzing other apps in these domains or apps from other domains might lead to different outcomes. However, the apps we included in our analysis were selected as they were the most popular in their application domains. Therefore, they are more likely to have completed their privacy labels and have higher quality privacy policies.

Internal validity threats might stem from the fact that human judgment was used to map privacy policy practices to privacy labels. Different judges might label the data claims in the apps' privacy policies differently or come up with different annotations. To mitigate this threat, the judges were provided with the exact definition of data types introduced by Apple. To minimize the error rate, the judges discussed the different data types before conducting the coding process to address their concerns and the results were crossed checked as well to enhance the integrity of the data.

4.8. Conclusion

In this chapter, we studied privacy labels in the Apple App Store. Our analysis was conducted using a dataset of 90 apps, sampled from the domains of ride-hailing, investing, and mental health. Our results showed that apps that are similar in terms of functionality can vastly differ in their data collection practices. In all of our investigated domains, there were several apps that deviate from the average privacy practices of their application domain, collecting either more or less data types than the average app in their domain. Our analysis has also revealed that, while there are some data types that are commonly collected by almost all apps in all domains, each domain has some data types that are specific to the domain. Examining the privacy policies of our apps has exposed some inconsistencies between the data collection claims in apps policies versus the data types

declared in the labels. In general, problems of under-reporting (claims appear in the policy but not the labels) and over reporting (data types that are declared in the labels but are not claimed in the privacy policy) were detected. It is not clear if these errors are a result of label misunderstanding or deceptive privacy practices.

Based on our findings, we propose different design strategies to enhance the effectiveness and utility of privacy labels in the app market. In particular, we suggest to architect the choice environment (app store) in such a way that can nudge users towards making more privacy-informed app selection decisions. Finally, based on our findings in this chapter, the following research directions will be pursued in our future work:

- We will conduct experiments to examine the influence of our suggested design recommendations on app users' behavior. Our objective is to test whether our suggested nudges would in fact influence users' decisions when they select which apps to download. We will further study the interaction effects between our suggested nudges and other factors that often influence app users' download decisions, such as app descriptions and screenshots.

- We will develop automated solutions for measuring the alignment between apps' privacy policies and their declared labels. These solutions will utilize text processing and modeling techniques to effectively identify data collection claims in apps policies and map them to privacy labels. Our objective is to help app developers avoid errors of under or over reporting.

Chapter 5. Conclusions

As of today, privacy enforcement remains a tremendous challenge for mobile app stores. App developers frequently deploy extreme privacy-invading tactics to collect users' data. App users, on the other hand, hardly take any sufficient measures to protect their information. This phenomenon is commonly known as the privacy paradox; or the discrepancy between people's privacy attitudes and their actual behaviors. In particular, due to a plethora of psychological distortions and cognitive biases, individuals are unlikely to act rationally when facing privacy-sensitive decisions, even when they perceive the risks to their privacy to be significant [193, 194].

The primary focus of this dissertation is the acknowledgment that users' privacy concerns and developers' data collection practices in the mobile app market are domain-specific. In other words, each domain of functionally-related apps has its own privacy paradox; what is acceptable in terms of privacy practices in one domain can be viewed as invasive or even malicious in another domain. Consequently, generic *one-size-fits-all* solutions that attempt to address privacy concerns across all domains are often insufficient in addressing the contextual characteristics of individual application domains. To overcome these limitations, this dissertation proposed the following contributions:

In **Chapter 2**, we proposed an automated approach for classifying mobile apps into more cohesive groups of functionally-related application domains. Currently, modern app stores enable developers to classify their apps by selecting from a set of static and generic categories, such as health, games, and music. However, with thousands of apps classified under each category, locating apps that match a specific consumer interest can be a challenging task. To overcome this challenge, we utilized word embeddings to create

numeric semantic representations of app descriptions, which were then classified to generate more cohesive categories of apps. Our aim was to enhance apps' visibility and discoverability as well as identify the functional boundaries of app collections. We evaluated our approach using a dataset of 600 apps sampled from the App Store categories of Education, Health&Fitness, and Medical. Our results showed that our classification algorithms performed best when app descriptions were vectorized using GloVe, a count-based word embedding model. We further conducted an experiment with the help of 12 human subjects to validate our approach. The results showed that our proposed approach produced app classifications that were closely aligned with our study participants' classifications.

In **Chapter 3** we proposed a domain-specific approach for summarizing privacy concerns in mobile app reviews. Our analysis was conducted using a dataset of 2.6 million app reviews sampled from three application domains. Our results revealed that users tend to express their privacy concerns using domain-specific vocabulary. This vocabulary can be leveraged to summarize users' privacy concerns in the domain. In particular, our evaluation showed that using domain-specific keywords can significantly improve the effectiveness of review summarization algorithms. We further showed that using the word embedding model GloVe to measure the semantic similarity between review words helped to reduce redundancy in the generated summaries. Our proposed approach is intended to help app developers quickly identify the critical privacy concerns in their domain of operation, and ultimately, modify their data collection practices accordingly to improve their apps' chances of survival.

In **Chapter 4** we conducted a detailed analysis of privacy labels in the Apple App Store. Using a dataset of 90 apps from the domains of ride-hailing, investing, and mental

health, we explored how apps expressed their data collection practices in their labels. Our analysis revealed that apps with similar core functionality can substantially vary in their data collection practices. We refer to apps that collect significantly less or significantly more data types than the average app in their domain as privacy outliers. Our analysis has also exposed several inconsistencies between apps' privacy labels and their privacy policies. These inconsistencies can take the form of over-reporting, where an app declares more data types in its labels than it reports in its policy, and under-reporting problems, where a data collection claim in the privacy policy of an app is not declared in its privacy labels. Based on our findings, we proposed several design strategies that can be used to improve the credibility and utility of privacy labels in app stores.

Bibliography

[1] Samuel Warren and Louis Brandeis. The right to privacy. *Harvard Law Review*, 4(5):193–220, 1890.

[2] Alan Westin and Oscar Ruebhausen. Privacy and freedom. *Washington and Lee Law Review*, 25:166, 1968.

[3] Daniel Solove. Conceptualizing privacy. *California Law Review*, 90:1087, 2002.

[4] Axe van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *IEEE International Symposium on Requirements Engineering*, pages 249–262, 2001.

[5] Abdulbaki Aydin, David Piorkowski, Omer Tripp, Pietro Ferrara, and Marco Pistoia. Visual configuration of mobile privacy policies. In *International Conference on Fundamental Approaches to Software Engineering*, pages 338–355, 2017.

[6] Li Li, Tegawendé Bissyandé, Mike Papadakis, Siegfried Rasthofer, Alexandre Bartel, Damien Octeau, Jacques Klein, and Le Traon. Static analysis of Android apps: A systematic literature review. *Information and Software Technology*, 88:67–95, 2017.

[7] Xiaoyin Wang, Xue Qin, Mitra Bokaei, Rocky Slavin, Travis Breaux, and Jianwei Niu. Guileak: Tracing privacy policy claims on user input data for Android applications. In *International Conference on Software Engineering*, pages 37–47, 2018.

[8] Alessandro Acquisti, Idris Adjerid, Rebecca Balebako, Laura Brandimarte, Lorrie Faith Cranor, Saranga Komanduri, Pedro Giovanni Leon, Norman Sadeh, Florian Schaub, Manya Sleeper, Yang Wang, and Shomir Wilson. Nudges for privacy and security: Understanding and assisting users' choices online. *ACM Computing Surveys*, 50(3):44, 2017.

[9] Elias Papadopoulos, Michalis Diamantaris, Panagiotis Papadopoulos, Thanasis Petsas, Sotiris Ioannidis, and Evangelos Markatos. The long-standing privacy debate: Mobile websites vs mobile apps. In *International Conference on World Wide Web*, pages 153–162, 2017.

[10] Fahimeh Ebrahimi, Miroslav Tushev, and Anas Mahmoud. Mobile app privacy in software engineering research: A systematic mapping study. *Information and Software Technology*, 133, 2020.

[11] Jaspreet Bhatia, Travis Breaux, and Florian Schaub. Mining privacy goals from privacy policies using hybridized task recomposition. *ACM Transactions on Software Engineering and Methodology*, 25(3):1–24, 2016.

[12] Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden, Alexandre Bartel,

Jacques Klein, Yves Traon, Damien Octeau, and Patrick McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. *Acm Sigplan Notices*, 49(6):259–269, 2014.

[13] Sebastian Zimmeck, Ziqi Wang, Lieyong Zou, Roger Iyengar, Bin Liu, Florian Shaub, Shomir Wilson, Norman Sadeh, Steven Bellovin, and Joel Reidenberg. Automated analysis of privacy requirements for mobile apps. In *AAAI Fall Symposium Series*, pages 286–296, 2016.

[14] Jessica Young. Commitment analysis to operationalize software requirements from privacy policies. *Requirements Engineering*, 16(1):33–46, 2011.

[15] Stuart McIlroy, Nasir Ali, Hammad Khalid, and Ahmed Hassan. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empirical Software Engineering*, 21(3):1067–1106, 2016.

[16] Hammad Khalid, Emad Shihab, Meiyappan Nagappan, and Ahmed Hassan. What do mobile app users complain about? *IEEE Software*, 32(3):70–77, 2015.

[17] Travis Breaux, Hanan Hibshi, and Ashwini Rao. Eddy, a formal language for specifying and analyzing data flow specifications for conflicting privacy requirements. *Requirements Engineering*, 19(3):281–307, 2014.

[18] Travis Breaux and Ashwini Rao. Formal analysis of privacy requirements specifications for multi-tier applications. In *International Requirements Engineering Conference*, pages 14–23, 2013.

[19] Yung Van Der Sype and Walid Maalej. On lawful disclosure of personal user data: What should app developers do? In *International Workshop on Requirements Engineering and Law*, pages 25–34, 2014.

[20] Inah Omoronyia, Luca Cavallaro, Mazeiar Salehie, Liliana Pasquale, and Bashar Nuseibeh. Engineering adaptive privacy: On the role of privacy awareness requirements. In *International Conference on Software Engineering*, pages 632–641, 2013.

[21] Keerthi Thomas, Arosha Bandara, Blaine Price, and Bashar Nuseibeh. Distilling privacy requirements for mobile applications. In *International Conference on Software Engineering*, pages 871–882, 2014.

[22] Phu Mai, Arda Goknil, Lwin Shar, Fabrizio Pastore, Lionel Briand, and Shaban Shaame. Modeling security and privacy requirements: A use case-driven approach. *Information and Software Technology*, 100:165–182, 2018.

[23] Yu Feng, Saswat Anand, Isil Dillig, and Alex Aiken. Apposcopy: Semantics-based detection of Android malware through static analysis. In *International Symposium*

on Foundations of Software Engineering, pages 576–587, 2014.

[24] Jianjun Huang, Xiangyu Zhang, Lin Tan, Peng Wang, and Bin Liang. Asdroid: Detecting stealthy behaviors in Android applications by user interface and program behavior contradiction. In *International Conference on Software Engineering*, pages 1036–1046, 2014.

[25] Alexandre Bartel, Jacques Klein, Martin Monperrus, and Yves Traon. Static analysis for extracting permission checks of a large scale framework: The challenges and solutions for analyzing Android. *IEEE Transactions on Software Engineering*, 40(6):617–632, 2014.

[26] Wei Xu, Fangfang Zhang, and Sencun Zhu. Permlyzer: Analyzing permission usage in Android applications. In *International Symposium on Software Reliability Engineering*, pages 400–410, 2013.

[27] Svitlana Vakulenko, Oliver Müller, and Jan Brocke. Enriching itunes app store categories via topic modeling. In *International Conference on Information Systems*, pages 1–11, 2014.

[28] Todd Cherner, Judy Dix, and Corey Lee. Cleaning up that mess: A framework for classifying educational apps. *Contemporary Issues in Technology and Teacher Education*, 14(2):158–193, 2014.

[29] Mobin Yasini and Guillaume Marchand. Toward a use case based classification of mobile health applications. In *Studies in health technology and informatics*, pages 175–179, 2015.

[30] Matthias Höhn, Ute Jan, Theodor Framke, and Urs-Vito Albrecht. Classification of health related applications. *Studies in health technology and informatics*, 266:139–142, 2016.

[31] Giacomo Berardi, Andrea Esuli, Tiziano Fagni, and Fabrizio Sebastiani. Multi-store metadata-based supervised mobile app classification. In *Annual ACM Symposium on Applied Computing*, pages 585–588, 2015.

[32] David Lulu and Tsvi Kuflik. Functionality-based clustering using short textual description: Helping users to find apps installed on their mobile device. In *International Conference on Intelligent User Interfaces*, pages 297–306, 2013.

[33] Shahab Mokarizadeh, Mohammad Rahman, and Mihhail Matskin. Mining and analysis of apps in google play. In *International Conference on Web Information Systems and Technologies*, pages 527–535, 2013.

[34] Afnan Al-Subaihin, Federica Sarro, Sue Black, Licia Capra, Mark Harman, Yue Jia,

and Yuanyuan Zhang. Clustering mobile apps based on mined textual features. In *International Symposium on Empirical Software Engineering and Measurement*, pages 1–38, 2016.

[35] Hengshu Zhu, Enhong Chen, Hui Xiong, Huanhuan Cao, and Jilei Tian. Mobile app classification with enriched contextual information. *IEEE Transactions on Mobile Computing*, 13(7):1550–1563, 2013.

[36] Borja Sanz, Igor Santos, Carlos Laorden, Xabier Ugarte-Pedrero, and Pablo Bringas. On the automatic categorisation of Android applications. In *Consumer Communications and Networking Conference*, pages 149–153, 2012.

[37] Rahul Pandita, Xusheng Xiao, Wei Yang, William Enck, and Tao Xie. WHYPER: Towards automating risk assessment of mobile applications. In *USENIX Security Symposium*, pages 527–542, 2013.

[38] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

[39] Maha Fraj, Mohamed Hajkacem, and Nadia Essoussi. A novel tweets clustering method using word embeddings. In *International Conference on Computer Systems and Applications*, pages 1–7, 2018.

[40] Hengshu Zhu, Huanhuan Cao, Enhong Chen, Hui Xiong, and Jilei Tian. Exploiting enriched contextual information for mobile app classification. In *International Conference on Information and Knowledge Management*, pages 1617–1621, 2012.

[41] Stephen Robertson and Steve Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *International Conference on Research and Development in Information Retrieval*, pages 345–354, 1994.

[42] David Blei, Andrew Ng, and Michael Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[43] Maleknaz Nayebi, Bram Adams, and Guenther Ruhe. Release practices for mobile apps–What do users and developers think? In *International Conference on Software Analysis, Evolution, and Reengineering*, pages 552–562, 2016.

[44] Afnan Al-Subaihin, Federica Sarro, Sue Black, and Licia Capra. Empirical comparison of text-based mobile apps similarity measurement techniques. In *Empirical Software Engineering*, pages 1–26, 2019.

[45] Anthony Finkelstein, Mark Harman, Yue Jia, William Martin, Federica Sarro, and Yuanyuan Zhang. Investigating the relationship between price, rating, and popular-

ity in the blackberry world app store. *Information and Software Technology*, 87:119–139, 2017.

[46] Alessandra Gorla, Ilaria Tavecchia, Florian Gross, and Andreas Zeller. Checking app behavior against app descriptions. In *International Conference on Software Engineering*, pages 1025–1035, 2014.

[47] Soo Lim, Peter Bentley, Natalie Kanakam, Fuyuki Ishikawa, and Shinichi Honiden. Investigating country differences in mobile app user behavior and challenges for software engineering. *IEEE Transactions on Software Engineering*, 41(1):40–64, 2015.

[48] Soo Ling Lim and Peter Bentley. Investigating app store ranking algorithms using a simulation of mobile app ecosystems. In *Congress on Evolutionary Computation*, pages 2672–2679, 2013.

[49] Cuiyun Gao, Jichuan Zeng, Zhiyuan Wen, David Lo, Xin Xia, Irwin King, and Michael Lyu. Emerging app issue identification via online joint sentiment-topic tracing. *arXiv preprint arXiv:2008.09976*, 2020.

[50] John Torous, Jennifer Nicholas, Mark E Larsen, Joseph Firth, and Helen Christensen. Clinical review of user engagement with mental health smartphone apps: evidence, theory and improvements. *Evidence-Based Mental Health*, 21(3):116–119, 2018.

[51] Laura Dennison, Leanne Morrison, Gemma Conway, and Lucy Yardley. Opportunities and challenges for smartphone applications in supporting health behavior change: Qualitative study. *Journal of Medical Internet Research*, 15(4):e86, 2013.

[52] Gunwood Lee and Raghu Santanam. Determinants of mobile apps' success: Evidence from the app store market. *Journal of Management Information Systems*, 31(2):133–170, 2014.

[53] Radim Rehurek and Petr Sojka. Software framework for topic modelling with large corpora. In *Workshop on New Challenges for NLP Frameworks*, 2010.

[54] Zellig Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[55] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Annual Meeting of the Association for Computational Linguistics*, pages 302–308, 2014.

[56] Moniba Keymanesh, Micha Elsner, and Srinivasan Parthasarathy. Toward domain-guided controllable summarization of privacy policies. In *Natural Legal Language Processing Workshop at KDD*, 2020.

[57] Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. Docbert: Bert

for document classification. *arXiv preprint arXiv:1904.08398*, 2019.

[58] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Workshop of 1st International Conference on Learning Representations*, 2013.

[59] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[60] Luis Gutiérrez and Brian Keith. A systematic literature review on word embeddings. In *International Conference on Software Process Improvement*, pages 132–141, 2018.

[61] Monte Hancock. *Data Mining: Supervised Learning*, pages 406–421. Taylor & Francis Group, 2016.

[62] Asaf Shabtai, Yuval Fledel, and Yuval Elovici. Automated static code analysis for classifying Android applications using machine learning. In *International Conference on Computational Intelligence and Security*, pages 329–333, 2010.

[63] William Martin, Mark Harman, Yue Jia, Federica Sarro, and Yuanyuan Zhang. The app sampling problem for app store mining. In *Conference on Mining Software Repositories*, pages 123–133, 2015.

[64] Claes Wohlin, Per Runeson, Martin Höst, Magnus Ohlsson, Björn Regnell, and Anders Wesslèn. *Experimentation in Software Engineering*. Springer, 2012.

[65] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European Conference on Machine Learning*, pages 137–142, 1998.

[66] Edward Loper and Steven Bird. NLTK: The natural language toolkit. In *COLING/ACL on Interactive Presentation Sessions*, pages 69–72, 2002.

[67] Martin Porter. An algorithm for suffix stripping. 14(3):130–137, 1980.

[68] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[69] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations*, 2016.

[70] Quoc Le and Tomas Mikolov. Distributed representations of sentences and docu-

ments. In *International conference on machine learning*, pages 1188–1196, 2014.

[71] Hani Safadi, Weifeng Li, Pouya Rahmati, Saber Soleymani, Krzysztof Kochut, and Amit Sheth. Curtailing fake news propagation with psychographics. *SSRN Electronic Journal*, 2020.

[72] Richard Socher, Eric Huang, Jeffrey Pennin, Christopher Manning, and Andrew Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in neural information processing systems*, pages 801–809, 2011.

[73] Amir Sadeghian and Alireza Sharafat. Bag of words meets bags of popcorn. 2015.

[74] Tom Kenter, Alexey Borisov, and Maarten Rijke. Siamese cbow: Optimizing word embeddings for sentence representations. *arXiv preprint arXiv:1606.04640*, 2016.

[75] Hinrich Schütze, Christopher Manning, and Prabhakar Raghavan. Introduction to information retrieval. In *International Communication of Association for Computing Machinery Conference*, 2008.

[76] Thomas Hofmann. Probabilistic latent semantic indexing. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 211–218, 2017.

[77] Thomas Griffiths and Mark Steyvers. Finding scientific topics. *National Academy of Sciences*, 101(1):5228–5235, 2004.

[78] Liangjie Hong and Brian Davison. Empirical study of topic modeling in Twitter. In *Workshop on Social Media Analytics*, pages 80–88, 2010.

[79] Ece Mutlu, Toktam Oghaz, Ege Tutunculer, and Ivan Garibay. Do bots have moral judgement? the difference between bots and humans in moral rhetoric. In *International Conference on Advances in Social Networks Analysis and Mining*, 2020.

[80] Pat Langley, Wayne Iba, and Kevin Thompson. An analysis of Bayesian classifiers. In *Artificial Intelligence*, pages 223–228, 1992.

[81] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[82] Tom Mitchell. *Machine learning*. McGraw-hill New York, 1997.

[83] Yoav Freund and Robert Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[84] David Kleinbaum and Mitchel Klein. *Logistic regression*. Springer, 2002.

[85] Emitza Guzman, Muhammad El-Haliby, and Bernd Bruegge. Ensemble methods for app review classification: An approach for software evolution (n). In *International Conference on Automated Software Engineering*, pages 771–776, 2015.

[86] Walid Maalej and Hadeer Nabil. Bug report, feature request, or simply praise? On automatically classifying app reviews. In *International Requirements Engineering Conference*, pages 116–125, 2015.

[87] Grant Williams, Miroslav Tushev, Fahimeh Ebrahimi, and Anas Mahmoud. Modeling user concerns in sharing economy: the case of food delivery apps. *Automated Software Engineering*, 27:229–263, 2020.

[88] Sida Wang and Christopher Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Annual Meeting of the Association for Computational Linguistics*, pages 90–94, 2012.

[89] Elizabeth Poché, Nishant Jha, Grant Williams, Jazmine Staten, Miles Vesper, and Anas Mahmoud. Analyzing user comments on youtube coding tutorial videos. In *International Conference on Program Comprehension*, pages 196–206, 2017.

[90] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[91] Wei Fu, Tim Menzies, and Xipeng Shen. Tuning for software analytics: Is it really necessary? *Information and Software Technology*, 76:135–146, 2016.

[92] Tuning the hyper-parameters of an estimator. `https://scikit-learn.org/stable/modules/grid_search.html`.

[93] Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of Documentation*, 2004.

[94] George Forman. A pitfall and solution in multi-class feature selection for text classification. In *International Conference on Machine Learning*, pages 38–46, 2004.

[95] Samuel Shapiro and Martin Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3):591–611, 1965.

[96] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*,

32(200):675–701, 1937.

[97] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979.

[98] Anthony Conger. Integration and generalization of kappas for multiple raters. *Psychological Bulletin*, 88(2):322, 1980.

[99] Yuan Tian, Meiyappan Nagappan, David Lo, and Ahmed Hassan. What are the characteristics of high-rated apps? A case study on free android applications. In *International Conference on Software Maintenance and Evolution*, pages 301–310, 2015.

[100] Robert Grissom and John Kim. *Effect sizes for research: A broad practical approach*. Lawrence Erlbaum Associates Publishers, 2005.

[101] Chris Martin. The sharing economy: A pathway to sustainability or a nightmarish form of neoliberal capitalism? *Ecological Economics*, 121:149–159, 2016.

[102] Tarik Dogru, Makarand Mody, and Courtney Suess. Adding evidence to the debate: Quantifying Airbnb's disruptive impact on ten key hotel markets. *Tourism Management*, 72:27–39, 2019.

[103] Giovanni Quattrone, Davide Proserpio, Daniele Quercia, Licia Capra, and Mirco Musolesi. Who benefits from the "sharing" economy of airbnb? In *International Conference on World Wide Web*, pages 1385–1394, 2016.

[104] Tawanna Dillahunt, Xinyi Wang, Earnest Wheeler, Hao Cheng, Brent Hecht, and Haiyi Zhu. The sharing economy in computing: A systematic literature review. *Proceedings of the ACM Human Computer Interaction*, 1:26, 2017.

[105] PwC. The sharing economy: Consumer intelligence series. *PricewaterhouseCoopers LLP*, 2015.

[106] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[107] Tore Dybå and Torgeir Dingsøyr. Strength of evidence in systematic reviews in software engineering. In *International Symposium on Empirical Software Engineering and Measurement*, pages 178–187, 2008.

[108] Jorge Pérez, Jessica Díaz, Javier Garcia-Martin, and Bernardo Tabuenca. Systematic literature reviews in software engineering—enhancement of the study selection process using cohen's kappa statistic. *Journal of Systems and Software*, 168, 2020.

[109] Chang Park and Hyun Kim. Measurement of inter-rater reliability in systematic review. *Hanyang Medical Reviews*, 35(1):44–49, 2015.

[110] He Jiang, Hongjing Ma, Zhilei Ren, Jingxuan Zhang, and Xiaochen Li. What makes a good app description? In *Asia-Pacific Symposium on Internetware*, pages 45–53, 2014.

[111] Lidong Bing, Wai Lam, and Tak-Lam Wong. Using query log and social tagging to refine queries based on latent topics. In *International Conference on Information and Knowledge Management*, pages 583–592, 2011.

[112] Liang Yu, Jin Wang, Robert Lai, and Xuejie Zhang. Refining word embeddings for sentiment analysis. In *Conference on Empirical Methods in Natural Language Processing*, pages 534–539, 2017.

[113] Marwa Naili, Anja Chaibi, and Henda Hajjami. Comparative study of word embedding methods in topic segmentation. *Procedia Computer Science*, 112:340–349, 2017.

[114] Miroslav Tushev, Fahimeh Ebrahimi, and Anas Mahmoud. Digital discrimination in sharing economy a requirements engineering perspective. In *International Requirements Engineering Conference*, pages 204–214, 2020.

[115] Miroslav Tushev, Fahimeh Ebrahimi, and Anas Mahmoud. Analysis of non-discrimination policies in sharing economy. In *International Conference on Software Maintenance and Evolution*, 2021.

[116] Axel van Lamsweerde. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, 2009.

[117] Tawanna Dillahunt and Amelia Malone. The promise of the sharing economy among disadvantaged communities. In *Annual ACM Conference on Human Factors in Computing Systems*, pages 2285–2294, 2015.

[118] Joshua West, Cougar Hall, Carl Hanson, Michael Barnes, Christophe Giraud-Carrier, and James Barrett. There's an app for that: Content analysis of paid health and fitness apps. *Journal of Medical Internet Research*, 14(3):e72, 2012.

[119] Pew Internet. Apps and privacy: More than half of app users have uninstalled or decided to not install an app due to concerns about their personal information. `https://www.pewresearch.org/internet/2012/09/05/privacy-and-data-management-on-mobile-devices-2/`, 2012.

[120] Omar Haggag, Sherif Haggag, John Grundy, and Mohamed Abdelrazek. Covid-19 vs social media apps: Does privacy really matter? In *International Conference on Software Engineering - Software Engineering in Society*, 2021.

[121] Jie Gu, Yunjie (Calvin) Xu, Heng Xu, Cheng Zhang, and Hong Ling. Privacy concerns for mobile app download: An elaboration likelihood model perspective. *Decision Support Systems*, 94:19–28, 2017.

[122] Majid Hatamian, Jetzabel Serna, and Kai Rannenberg. Revealing the unrevealed: Mining smartphone users privacy perception on app markets. *Computers & Security*, 83:332–353, 2019.

[123] Adelina Ciurumelea, Andreas Schaufelbühl, Sebastiano Panichella, and Harald Gall. Analyzing reviews and code of mobile apps for better release planning. In *International Conference on Software Analysis, Evolution and Reengineering*, pages 91–102, 2017.

[124] Ning Chen, Jialiu Lin, Steven Hoi, Xiaokui Xiao, and Boshen Zhang. Ar-miner: mining informative reviews for developers from mobile app marketplace. In *International Conference on Software Engineering*, pages 767–778, 2014.

[125] Andrea Sorbo, Sebastiano Panichella, Carol Alexandru, Junji Shimagaki, Corrado Visaggio, Gerardo Canfora, and Harald Gall. What would users change in my app? Summarizing app reviews for recommending software changes. In *International Symposium on Foundations of Software Engineering*, pages 499–510, 2016.

[126] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado Visaggio, Canfora Aaron Gerardo, and Harald Gall. How can I improve my app? Classifying user reviews for software maintenance and evolution. In *International Conference on Software Maintenance and Evolution*, pages 281–290, 2015.

[127] Zijad Kurtanović and Walid Maalej. Mining user rationale from software reviews. In *International Requirements Engineering Conference*, pages 61–70, 2017.

[128] Andrew Besmer, Jason Watson, and Shane Banks. Investigating user perceptions of mobile app privacy: An analysis of user-submitted app reviews. *International Journal of Information Security and Privacy*, 14(4):74–91, 2020.

[129] Debjyoti Mukherjee, Alireza Ahmadi, Maryam VahdatPour, and Joel Reardon. An empirical study on user reviews targeting mobile apps' security & privacy. *arXiv preprint arXiv:2010.06371*, 2020.

[130] Duc Nguyen, Erik Derr, Michael Backes, and Sven Bugiel. Short text, large effect: Measuring the impact of user reviews on android app security & privacy. In *Symposium on Security and Privacy*, pages 555–569, 2019.

[131] William Martin, Federica Sarro, Yue Jia, Yuanyuan Zhang, and Mark Harman. A survey of app store analysis for software engineering. *IEEE Transactions on Software Engineering*, 43(9):817–847, 2017.

[132] Hanyang Hu, Shaowei Wang, Cor-Paul Bezemer, and Ahmed Hassan. Studying the consistency of star ratings and reviews of popular free hybrid android and ios apps. *Empirical Software Engineering*, 24(1):7–32, 2019.

[133] Rajesh Vasa, Leonard Hoon, Kon Mouzakis, and Akihiro Noguchi. A preliminary analysis of mobile app user reviews. In *Computer-Human Interaction Conference*, pages 241–244, 2012.

[134] Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. A biterm topic model for short texts. In *International Conference on World Wide Web*, pages 1445–1456, 2013.

[135] PYMNTS. High-speed traders pay robinhood $331 million in q1 to execute trades. `https://www.pymnts.com/earnings/2021/high-speed-traders-pay-robinhood-331-million-dollars-q1-execute-trades/`, 2021.

[136] Levi Sumagaysay. The pandemic has more than doubled food-delivery apps business. now what? `https://www.marketwatch.com.`, 2020.

[137] Robert Longyear and Kostadin Kushlev. Can mental health apps be effective for depression, anxiety, and stress during a pandemic? *Practice Innovations*, 6(2):131–137, 2021.

[138] Lisa Cosgrove, Justin Karter, and Zenobia Morrill. Psychology and surveillance capitalism: The risk of pushing mental health apps during the covid-19 pandemic. *Journal of Humanistic Psychology*, 60(5):611–625, 2020.

[139] Leonard Hoon, Rajesh Vasa, Jean-Guy Schneider, and Kon Mouzakis. A preliminary analysis of vocabulary in mobile app user reviews. In *Australian Computer-Human Interaction Conference*, pages 245–248, 2012.

[140] Gerlof Bouma. Normalized (pointwise) mutual information in collocation extraction. *German Society for Computational Linguistics*, 30:31–40, 2009.

[141] Miroslav Tushev, Fahimeh Ebrahimi, and Anas Mahmoud. Domain-specific analysis of mobile app reviews using keyword-assisted topic models. In *International Conference on Software Engineering*, 2022.

[142] Andrea Sorbo, Giovanni Grano, Corrado Aaron, and Sebastiano Panichella. Investigating the criticality of user-reported issues through their relations with app rating. *Journal of Software: Evolution and Process*, 33(3):e2316, 2021.

[143] Clare Llewellyn, Claire Grover, and Jon Oberlander. Summarizing newspaper comments. In *International Advancement of Artificial Intelligence Conference on We-*

blogs and Social Media, pages 599—-602, 2014.

[144] Elham Khabiri, James Caverlee, and Chiao-Fang Hsu. Summarizing user-contributed comments. In *International Advancement of Artificial Intelligence Conference on Weblogs and Social Media*, 2011.

[145] Nishant Jha and Anas Mahmoud. Using frame semantics for classifying and summarizing application store reviews. *Empirical Software Engineering*, 23(6):3734–3767, 2018.

[146] Udo Hahn and Inderjeet Mani. The challenges of automatic summarization. *Computer*, 33(11):29–36, 2000.

[147] Lauren Squires. Enregistering internet language. *2010*, 39(4):457–492, Language in society.

[148] Jackie Cheung. Comparing abstractive and extractive summarization of evaluative text: controversiality and content selection. *Thesis in the Department of Computer Science of the Faculty of Science, University of British Columbia*, 47, 2008.

[149] David Inouye and Jugal Kalita. Comparing twitter summarization algorithms for multiple post summaries. In *International Conference on Privacy, Security, Risk and Trust and International Conference on Social Computing*, pages 298–306, 2011.

[150] Anas Mahmoud and Grant Williams. Detecting, classifying, and tracing non-functional software requirements. *Requirements Engineering*, 21(3):357–381, 2016.

[151] Jaspreet Bhatia and Travis Breaux. Semantic incompleteness in privacy policy goals. In *International Requirements Engineering Conference*, pages 159–169, 2018.

[152] Yizhaq Benbenisty, Irit Hadar, Gil Luria, and Paola Spoletini. Privacy as first-class requirements in software development: A socio-technical approach. In *IEEE/ACM International Conference on Automated Software Engineering*, pages 1363–1367, 2021.

[153] Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. Weakly-supervised neural text classification. In *ACM International Conference on Information and Knowledge Management*, page 983–992, 2018.

[154] Xiaodong Gu and Sunghun Kim. What parts of your apps are loved by users? In *International Conference on Automated Software Engineering*, page 760–770, 2015.

[155] Maleknaz Nayebi, Homayoon Farrahi, Ada Lee, Henry Cho, and Guenther Ruhe. More insight from being more focused: Analysis of clustered market apps. In *International Workshop on App Market Analytics*, pages 30–36, 2016.

[156] Surekha Borra. *COVID-19 Apps: Privacy and Security Concerns*. Springer Singapore, 2020.

[157] Muhammad Azad, Junaid Arshad, Muhammad Akmal, Farhan Riaz, Sidrah Abdullah, Muhammad Imran, and Farhan Ahmad. A first look at privacy analysis of covid-19 contact-tracing mobile applications. *IEEE Internet of Things Journal*, 8(21):15796–15806, 2021.

[158] Ramona Trestian, Guodong Xie, Pintu Lohar, Edoardo Celeste, Malika Bendechache, Rob Brennan, Evgeniia Jayasekera, Regina Connolly, and Irina Tal. Privacy in a time of covid-19: How concerned are you? *IEEE Security and Privacy*, 19(5):26–35, 2021.

[159] Stuart Mcilroy, Weiyi Shang, Nasir Ali, and Ahmed Hassan. User reviews of top mobile apps in apple and google app stores. *Communications of the ACM*, 60(11):62–67, 2017.

[160] Hui Guo and Munindar Singh. Caspar: extracting and synthesizing user stories of problems from app reviews. In *International Conference on Software Engineering*, pages 628–640, 2020.

[161] Andrea Sorbo, Sebastiano Panichella, Corrado Visaggio, Massimiliano Penta, Gerardo Canfora, and Harald Gall. Exploiting natural language structures in software informal documentation. *IEEE Transactions on Software Engineering*, 47(8):1587–1604, 2021.

[162] Jaspreet Bhatia, Morgan Evans, and Travis Breaux. Identifying incompleteness in privacy policy goals using semantic frames. *Requirements Engineering*, 24(3):291–313, 2019.

[163] Ali Balapour, Hamid Reza Nikkhah, and Rajiv Sabherwal. Mobile application security: Role of perceived privacy as the predictor of security perceptions. *International Journal of Information Management*, 52:102063, 2020.

[164] Grant Williams and Anas Mahmoud. Modeling user concerns in the app store: A case study on the rise and fall of Yik Yak. In *International Requirements Engineering Conference*, pages 64–75, 2018.

[165] Patrick Kelley, Joanna Bresee, Lorrie Cranor, and Robert Reeder. A "nutrition label" for privacy. In *The Symposium on Usable Privacy and Security*, 2009.

[166] Patrick Kelley, Lorrie Cranor, and Norman Sadeh. Privacy as part of the app decision-making process. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 3393–3402, 2013.

[167] Tianshi Li, Kayla Reiman, Yuvraj Agarwal, Lorrie Cranor, and Jason Hong. Understanding challenges for developers to create accurate privacy nutrition labels. In *CHI Conference on Human Factors in Computing Systems*, pages 1–24, 2022.

[168] Yucheng Li, Deyuan Chen, Tianshi Li, Yuvraj Agarwal, Lorrie Cranor, and Jason Hong. Understanding ios privacy nutrition labels: An exploratory large-scale analysis of app store data. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pages 1–7, 2022.

[169] Yue Xiao, Zhengyi Li, Yue Qin, Jiale Guan, Xiaolong Bai, Xiaojing Liao, and Luyi Xing. Lalaine: Measuring and characterizing non-compliance of apple privacy labels at scale. *arXiv preprint arXiv:2206.06274*, 2022.

[170] Richard Thaler and Cass Sunstein. *Nudge: Improving Decisions About Health, Wealth, and Happiness Paperback*. Penguin Books, 2009.

[171] Markus Weinmann, Christoph Schneider, and Jan vom Brocke. Digital nudging. *Business & Information Systems Engineering*, 58(6):433–436, 2016.

[172] Brishen Rogers. The social costs of Uber. 82(6), 2015.

[173] Brenton Malin and Curry Chandler. Free to work anxiously: Splintering precarity among drivers for Uber and Lyft. *Communication, Culture and Critique*, 10(2):382–400, 2017.

[174] Christoph Lutz, Christian Hoffmann, Eliane Bucher, and Christian Fieseler. The role of privacy concerns in the sharing economy. *Information, Communication and Society*, 21(10):1472–1492, 2018.

[175] Giulia Ranzini, Michael Etter, Christoph Lutz, and Ivar Vermeulen. Privacy in the sharing economy. *SSRN Electronic Journal*, 2017.

[176] Darren Hayes, Christopher Snow, and Saleh Altuwayjiri. Geolocation tracking and privacy issues associated with the uber mobile application. In *The Conference on Information Systems Applied Research*, 2017.

[177] Sultan Al-Masaeed, Mohammad Al Nawayseh, Bader AlFawwaz, Mahmoud Maqableh, Mohammad Alnabhan, Raed Masadeh, and Atallah AL-Shatnawi. Factors affecting consumers' intention to use mobile ride hailing services in developing countries. *International Journal of Interactive Mobile Technologies*, 16(11):207–223, 2022.

[178] David Bakker, Nikolaos Kazantzis, Debra Rickwood, and Nikki Rickard. Mental health smartphone apps: review and evidence-based recommendations for future developments. *JMIR Mental Health*, 3(1), 2016.

[179] Julie Robillard, Tanya Feng, Arlo Sporn, Jen-Ai Lai, Cody Lo, Monica Ta, and Roland Nadler. Availability, readability, and content of privacy policies and terms of agreements of mental health apps. *Internet Interventions*, 17:100243, 2019.

[180] Tobias Dehling, Fangjian Gao, Stephan Schneider, and Ali Sunyaev. Exploring the far side of mobile health: Information security and privacy of mobile health apps on ios and android. *JMIR mHealth uHealth*, 3(1):e8, 2015.

[181] Sayan Chaudhry and Chinmay Kulkarni. Design patterns of investing apps and their effects on investing behaviors. In *Designing Interactive Systems Conference*, pages 777–788, 2021.

[182] Suzanee Malhotra. Study of features of mobile trading apps : A silver lining of pandemic. *Journal of Global Information and Business Strategy*, 12(1):75–80, 2020.

[183] Ayoub Majjodi, Alain Starke, and Christoph Trattner. Nudging towards health? examining the merits of nutrition labels and personalization in a recipe recommender system. In *The ACM Conference on User Modeling, Adaptation and Personalization*, pages 48–56, 2022.

[184] Catherine Cioffi, David Levitsky, Carly Pacanowski, and Fredrik Bertz. A nudge in a healthy direction. the effect of nutrition labels on food purchasing behaviors in university dining facilities. *Appetite*, 92:7–14, 2015.

[185] C A Roberto and N Khandpur. Improving the design of nutrition labels to promote healthier food choices and reasonable portion sizes. *International Journal of Obesity*, 38(1):25–33, 2014.

[186] Romain Cadario and Pierre Chandon. Which healthy eating nudges work best? a meta-analysis of field experiments. *Marketing Science*, 39(3):465–486, 2020.

[187] Siegwart Lindenberg and Esther Papies. Two kinds of nudging and the power of cues: Shifting salience of alternatives and shifting salience of goals. *International Review of Environmental and Resource Economics*, 13:229–263, 2019.

[188] Fahimeh Ebrahimi, Miroslav Tushev, and Anas Mahmoud. Classifying mobile applications using word embeddings. *ACM Transactions on Software Engineering and Methodology*, 31(2), 2021.

[189] Marta Santos Silva. Nutri-score as a nudging technique to enhance healthier food choices, 2021.

[190] Desiree Hagmann and Michael Siegrist. Nutri-score, multiple traffic light and incomplete nutrition labelling on food packages: Effects on consumers' accuracy in identifying healthier snack options. *Food Quality and Preference*, 83:103894, 2020.

[191] Stephanie Mertens, Mario Herberz, Ulf Hahnel, and Tobias Brosch. The effectiveness of nudging: A meta-analysis of choice architecture interventions across behavioral domains. *Proceedings of the National Academy of Sciences*, 119(1):e2107346118, 2022.

[192] Xusheng Xiao, Nikolai Tillmann, Manuel Fahndrich, Jonathan Halleux, Michal Moskal, and Tao Xie. User-aware privacy control via extended static-information-flow analysis. *Automated Software Engineering*, 22(3):333–366, 2015.

[193] Alessandro Acquisti. Privacy in electronic commerce and the economics of immediate gratification. In *ACM Conference on Electronic Commerce*, pages 21–29, 2004.

[194] Alessandro Acquisti, Laura Brandimarte, and George Loewenstein. Privacy and human behavior in the age of information. *Science*, 347(6221):509–514, 2015.

Vita

Fahimeh Ebrahimi Meymand is a PhD in the Department of Computer Science and Engineering at Louisiana State University. Her research focuses on extracting privacy concerns and practices in the mobile application market. During her time at Louisiana State University, Fahimeh has published several articles in top-tier academic journals and presented her research at numerous international conferences. Prior to her doctoral studies, Fahimeh worked as a software engineer at a leading technology company. She holds a Bachelor's degree and a Master's degree in Information Technology Engineering from Amirkabir University of Technology (Tehran Polytechnic).