ATLAS: Automatically Detecting Discrepancies Between Privacy Policies and Privacy Labels

Akshath Jain

CMU-CS-23-105
March 2023

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Norman Sadeh, Chair
Eunsuk Kang

Submitted in partial fulfillment of the requirements for the degree of Master of Science.

Keywords: Natural Language Processing, Machine Learning, Transformers, Privacy Policies, Privacy Labels, iOS

Abstract

Privacy policies are long, complex documents that end-users seldom read. Privacy labels aim to ameliorate these issues by providing succinct summaries of salient data practices. In December 2020, Apple began requiring that app developers submit privacy labels describing their apps' data practices. Yet, research suggests that app developers often struggle to do so. In this paper, we automatically identify possible discrepancies between mobile app privacy policies and their privacy labels. Such discrepancies could be indicators of potential privacy compliance issues.

We introduce the Automated Privacy Label Analysis System (ATLAS). ATLAS includes three components: a pipeline to systematically retrieve iOS App Store listings and privacy policies; an ensemble-based classifier capable of predicting privacy labels from the text of privacy policies with 91.3% accuracy using state-of-the-art NLP techniques; and a discrepancy analysis mechanism that enables a large-scale privacy analysis of the iOS App Store.

Our system has enabled us to analyze 354,725 iOS apps. We find several interesting trends. For example, only 40.3% of apps in the App Store provide easily accessible privacy policies, and only 29.6% of apps provide both accessible privacy policies and privacy labels. Among apps that provide both, 88.0% have at least one possible discrepancy between the text of their privacy policy and their privacy label, which could be indicative of a potential compliance issue. We find that, on average, apps have 5.32 such potential compliance issues.

We hope that ATLAS will help app developers, researchers, regulators, and mobile app stores alike. For example, app developers could use our classifier to check for discrepancies between their privacy policies and privacy labels, and regulators could use our system to help review apps at scale for potential compliance issues.

Acknowledgments

I would like to thank my advisor, Norman Sadeh, for providing guidance with this research project over the course of the program. I have been a part of Norman's lab since the spring semester of my first year at Carnegie Mellon (so a little over 4 years now), and he has been a huge influence in helping me become a well-rounded researcher.

I would also like to thank Yuanyuan Feng and Yaxing Yao for their invaluable mentorship over these past few years. Both of you have helped me greatly, and it would not have been possible for me to get here without you.

Additionally, I would like to thank Eunsuk Kang, for taking time out of his schedule to serve on my thesis committee and provide feedback on this thesis.

Finally, thank you to my friends and family – you have all provided unwavering support, and I could not have done it without you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Notice is a cornerstone of privacy: entities collecting information are expected to disclose the types of data collected, and how it is used. Privacy policies serve as the primary mechanism for notice, yet research has shown that privacy policies are long, complex documents that users seldom read [28] [36] [35].

Privacy labels aim to ameliorate this issue by providing succinct descriptions of salient data practices in an easy to consume format. In December 2020, Apple began requiring that developers include privacy labels for the apps they publish on the iOS App Store. However, recent research suggests that mobile app developers often struggle to understand and disclose their data practices. [24] [44].

In this work, we provide a detailed analysis of privacy policies and privacy labels in the iOS App Store by analyzing 354,725 iOS apps. We explore the state of privacy in the App Store, develop an ensemble-based classifier to automatically generate privacy labels from the text of privacy policies, and we provide a thorough compliance analysis by comparing the text of privacy policies to privacy labels.

1.1 Research Questions

In this paper, we pose three categories of research questions concerning privacy policies and privacy labels in the United States iOS App Store. These questions guide us in creating the Automated Privacy Label Analysis System (ATLAS). Details and capabilities of ATLAS are discussed in the next section.

1. What is the state of privacy policy and privacy label adoption among iOS apps? Can app privacy policies be easily accessed – specifically, how many apps have direct links to their privacy policies in the App Store? What percentage of apps have privacy labels? What percentage have both?
2. Is it possible to predict privacy labels from the text of privacy policies? Prior research indicates that developers struggle to create accurate labels. As a result, privacy labels may not always reflect true data collection [24], [22]. So, despite privacy labels being noisy, is it possible to train reliable classifiers?

3. Finally, are privacy labels consistent with the text of privacy policies? What types of discrepancies occur between privacy policies and labels? How many discrepancies do apps have on average? Is there a correlation between these rates and the popularity of mobile apps?

1.2 Key Contributions

To enable us to answer the research questions posed in the prior section, we developed the Automated Privacy Label Analysis System (ATLAS). Our system analyzed iOS app metadata, privacy policies, and privacy labels. It then flagged instances where privacy labels were not consistent with the text of their privacy policies, which we characterized as potential compliance issues. This was done using state-of-the-art natural language processing techniques involving unsupervised and supervised machine learning. ATLAS was designed to be highly scalable, and has enabled us to analyze 354,725 iOS apps. This paper makes several key contributions:

1. A scalable pipeline that enables systematically scraping iOS metadata, including privacy policies URLs and privacy labels. We include a machine learning model that determines if the app's privacy policy URL actually leads to an English-language privacy policy. We also include tools to download the text of the policy.
2. An ensemble-based classifier that can effectively generate privacy labels from the text of privacy policies. We formulated this as a multi-class, multi-label document classification problem. This enabled our classifier to identify if a privacy policy "Collects" or "Does Not Collect" a data type, for 32 separate data types.
3. A privacy analysis of the iOS App Store. We provided an extensive analysis of data practice disclosure discrepancies between the text of privacy policies and their corresponding privacy labels in the iOS App Store. We also include metrics such as privacy policy accessibility, privacy label adoption, and potential compliance issue existence.

We hope that ATLAS will help app developers, researchers, regulators, and mobile app stores alike. For example, app developers could use our classifier to check for discrepancies between their privacy policies and privacy labels. Meanwhile, app store operators and regulators could use our system to monitor discrepancy trends to effectively focus efforts on apps likely to have potential compliance issues.

1.3 Outline

Chapter 2 provides additional background information on privacy labels, as well as a comprehensive overview of related literature. Chapter 3 discusses the data collection pipeline that ATLAS uses, which enabled us to answer the first set of research questions. Chapter 4 describes the methodology used for training a set of classifiers used to generate privacy labels from the text of privacy policies, enabling us to answer the second set of research questions. Chapter 5 discusses the consistency of privacy policies and their privacy labels, enabling us to answer the third set of

research questions. Chapter 6 provides a detailed discussion of our results. Chapter 7 provides guidance for future work. And finally, Chapter 8 concludes this document.

Chapter 2

Background and Related Work

2.1 Privacy Labels

2.1.1 History of Privacy Labels

Privacy policies are long, complex documents that end-users seldom read. Seminal work in 2008 by McDonald et al demonstrated that reading every privacy policy a user encountered in a year would take approximately 244 hours – an impractical amount [28]. Attempts to make privacy policies more digestible included P3P – a method for websites to convey their privacy policies in a machine readable format – and P3P Expandable Grid – a user-agent program designed to assist users in viewing P3P compatible policies [9] [34]. However, the P3P standard was only loosely adopted, due to the lack of limited functionality provided by widely available user-agents and complexity of the standard [10]. Multilayered Privacy Policies, such as those proposed by the law firm Hunton & William LLP in conjunction with The Center for Information Policy Leadership, included summaries with standardized section headings; however, section content still contained free-form legal text [27].

To address shortcomings of prior approaches, Kelley et al. developed the "Privacy Label" to succinctly describe privacy policies in 2009, akin to nutritional labels found on food in 2009 [18]. The approach was largely successful, as research released the next year conducted a user study that found privacy labels had significant positive effects on accuracy, information retrieval speed, and reader enjoyment with privacy policies [19].

With the burgeoning mobile application markets on iOS and Android in 2013, further work by Kelley et al. proposed bringing privacy labels to mobile app stores [20]. The authors conducted a user study where participants could view a "Privacy Checklist" for an app before downloading it; they found that the inclusion of such a checklist affected user decisions in downloading apps, "especially when choosing between otherwise similar apps" [20].

2.1.2 iOS Privacy Labels

In December 2020, Apple began requiring all developers include "App Privacy Details" describing their apps' data collection practices when uploading new versions of their apps, arguably the largest adoption of privacy labels to date [16]. App Privacy Details are included within an app's

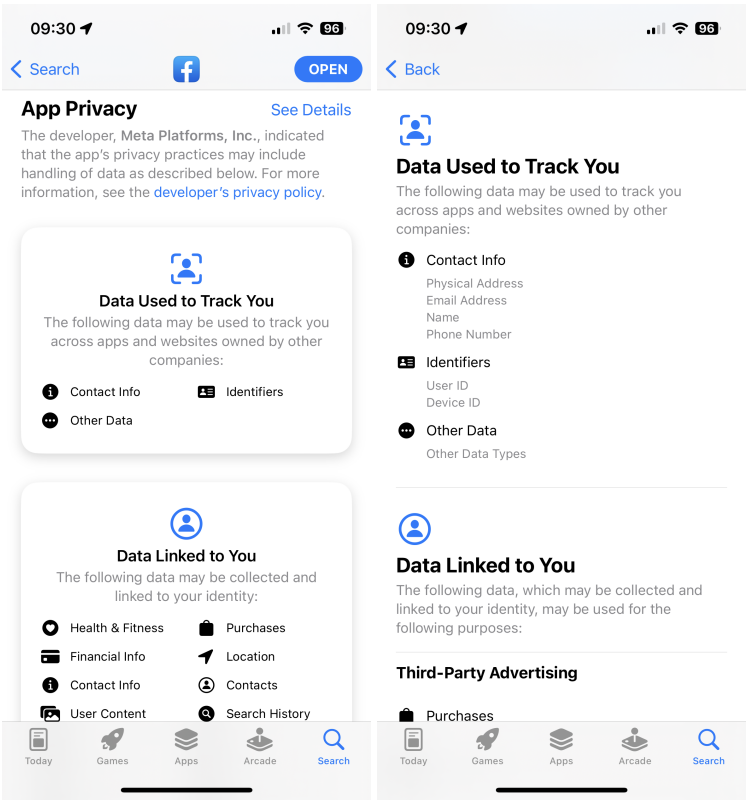


Figure 2.1: An example of a privacy label within the iOS App Store. The image on the left shows an overview of the app's privacy label. The image on the right depicts a more detailed summary of the app's data collection.

listing in the iOS App Store, enabling users to view data collection practices before downloading an app. The implementation is largely similar to the model proposed by researchers in [20].

Figure 2.1 shows an example of an iOS privacy label available within the iOS App Store. Apple takes a multilayered approach to privacy labels: users first encounter a summarized version, but can choose to "See Details," which provides a more comprehensive overview of the app's data collection. Developers are required to report data collected from the user, which refers to any data that is transmitted "off the device in a way that allows [developers] or third-party partners to access [the data] for a period longer than what is necessary to service the transmitted request in real time" [16].

The iOS privacy label consists of four parts. First, developers declare the data type(s) being collected by their app, as summarized in Table 2.1. Then, for each data type, developers are required to report how the data type is being used, as summarized in Table 2.2. Next, developers must specify if the data type is "Linked to You," which indicates that it is being collected non-anonymously. Finally, if the developer declares that the data type is "Linked to You," then they must declare if it is "Used to Track You," which indicates linking collected data "from your app about a particular end-user or device, such as a user ID, device ID, or profile, with Third-Party Data for targeted advertising or advertising measurement purposes, or sharing data collected from your app about a particular end-user or device with a data broker" [16].

Table 2.1: The following table summarizes the data types that an app can collect and report on its privacy label.

Data Type	Description
Name	Such as first or last name
Email Address	Including but not limited to a hashed email address
Phone Number	Including but not limited to a hashed phone number
Physical Address	Such as home address, physical address, or mailing address
Other User Contact	Info Any other information that can be used to contact the user outside the app
Health	Health and medical data, including but not limited to data from the Clinical Health Records API, HealthKit API, MovementDisorderAPIs, or health-related human subject research or any other user provided health or medical data
Fitness	Fitness and exercise data, including but not limited to the Motion and Fitness API
Payment Info	Such as form of payment, payment card number, or bank account number. If your app uses a payment service, the payment information is entered outside your app, and you as the developer never have access to the payment information, it is not collected and does not need to be disclosed.
Credit Info	Such as credit score
Other Financial Info	Such as salary, income, assets, debts, or any other financial information
Precise Location	Information that describes the location of a user or device with the same or greater resolution as a latitude and longitude with three or more decimal places
Coarse Location	Information that describes the location of a user or device with lower resolution than a latitude and longitude with three or more decimal places, such as Approximate Location Services
Sensitive Info	Such as racial or ethnic data, sexual orientation, pregnancy or childbirth information, disability, religious or philosophical beliefs, trade union membership, political opinion, genetic information, or biometric data
Contacts	Such as a list of contacts in the user's phone, address book, or social graph
Emails or Text Messages	Including subject line, sender, recipients, and contents of the email or message
Photos or Videos	The user's photos or videos
Audio Data	The user's voice or sound recordings
Gameplay Content	Such as saved games, multiplayer matching or gameplay logic, or user-generated content in-game
Customer Support	Data generated by the user during a customer support request
Other User Content	Any other user-generated content
Browsing History	Information about content the user has viewed that is not part of the app, such as websites
Search History	Information about searches performed in the app
User ID	Such as screen name, handle, account ID, assigned user ID, customer number, or other user- or account-level ID that can be used to identify a particular user or account

Device ID	Such as the device's advertising identifier, or other device-level ID
Purchase History	An account's or individual's purchases or purchase tendencies
Product Interaction	Such as app launches, taps, clicks, scrolling information, music listening data, video views, saved place in a game, video, or song, or other information about how the user interacts with the app
Advertising Data	Such as information about the advertisements the user has seen
Other Usage Data	Any other data about user activity in the app
Crash Data	Such as crash logs
Performance Data	Such as launch time, hang rate, or energy use
Other Diagnostic Data	Any other data collected for the purposes of measuring technical diagnostics related to the app
Other Data Types	Any other data types not mentioned

Table 2.2: Developers must also report how each data type is being used, by specifying one or more purposes.

Purpose	**Definition**
Third-Party Advertising	Such as displaying third-party ads in your app, or sharing data with entities who display third-party ads
Developer's Advertising or Marketing	Such as displaying first-party ads in your app, sending marketing communications directly to your users, or sharing data with entities who will display your ads
Analytics	Using data to evaluate user behavior, including to understand the effectiveness of existing product features, plan new features, or measure audience size or characteristics
Product Personalization	Customizing what the user sees, such as a list of recommended products, posts, or suggestions
App Functionality	Such as to authenticate the user, enable features, prevent fraud, implement security measures, ensure server up-time, minimize app crashes, improve scalability and performance, or perform customer support
Other Purposes	Any other purposes not listed

2.1.3 Issues with iOS Privacy Labels

While iOS privacy labels are intended to help inform end-users about apps' data practices, they are not without issue. Recent research suggests that developers face challenges in creating accurate privacy labels [24] [15]. After conducting interviews with iOS developers, the authors found that misconceptions about privacy labels often resulted in under- or over-reporting data collection [24]. Koch et al. conducted an "exploratory statistical evaluation" of 11,074 iOS apps, and found that only a "small number of apps provide privacy labels" [22]. They also conducted a dynamic analysis of a small subset of 1,687 apps, finding that 276 (16%) violated their own privacy labels by transmitting data without declaration [22]. This work expands on Koch et al. by conducting a broader statistical evaluation on a much larger dataset, and by investigating to

what extent privacy labels are inconsistent with the text of their privacy policies.

2.1.4 Generating iOS Privacy Labels

While iOS privacy labels have several issues, Li et al. suggest that auto-generating privacy labels might help increase their accuracy [24]. Tools such as PrivacyFlash Pro, and its successor Privacy Label Wiz, aid developers in creating privacy policies and privacy labels for iOS apps [47] [15]. These tools statically analyze iOS app source code for signatures such as plist permission strings, import statements, and class instantiations. These code signatures are then used as evidence of privacy practices, which are then converted into privacy policies and iOS privacy labels [50] [15]. Importantly, these tools can help developers comply with regulations, such as GDPR, which require accurate privacy notices [48].

Whereas Gardner et al. only focused on a small subset of iOS privacy labels (9 of 32), this work aims to analyze data collection disclosure for all 32 data types [15]. Moreover, while [50] and [15] used static code analysis, this work uses state-of-the-art natural language processing techniques to generate privacy labels directly from the text of privacy policies and compare the predicted labels with those reported within the App Store.

2.2 Applicable Legislation

Regulatory requirements for privacy disclosures vary by jurisdiction. However, the state of California and the European Union – two of the largest digital markets – have stringent privacy disclosure requirements. California requires compliance through the CCPA and the European Union through GDPR. Failure to provide accurate privacy disclosures in both jurisdictions could have significant legal ramifications.

2.3 Automated Mobile App Privacy Analyses

With millions of mobile apps, automation is the only way to analyze privacy practices at scale. Automating this type of analysis has been extensively studied. Early work by Enck et al. proposed TaintDroid, an instrumented version of Android that analyzes potential misuses of user data in realtime [13]. The researchers discovered 68 potential instances of user data being misued across 20 of the 30 apps analyzed [13]. Dynamic analysis systems, such as TaintDroid, actively run apps and monitor their behavior [13] [33]. While this gives a true representation of an app's behavior, it has significant overhead – limiting scale. Dynamic analysis on mobile platforms is also largely limited to Android, as the iOS operating system is closed-source and unable to be instrumented, though recent systems have focused on iOS as well [42].

A larger analysis of 17,991 Android apps was conducted by Zimmeck et al. by comparing privacy policies to static analyses of apps [47]. Static analysis was done by first decompiling apps [47]. Then, the researchers looked at permissions and call graphs to determine what types of user data were accessed and used by those apps [47]. The researchers found, that on average,

apps exhibited 1.83 potential privacy requirement inconsistencies – that is, apps tended to not disclose some types of behaviors in their policies [47].

A later study by Zimmeck et al. drastically increased the number of analyzed Android apps to over one million [49]. In their study, the authors found that 49.1% of apps do not have privacy policies; however, statically analyzing decompiled app binaries indicated that 88.6% of apps engage in behaviors requiring policies. Moreover, the researchers found an average of 2.89 potential compliance issues per app.

While the majority of studies have focused on the Android operating system, more recent research has been conducted on iOS. Kollnig et al. found that Apple's recent change to require user permission for apps to access device identifiers (i.e. the IDFA), combined with increased transparency through the use of privacy labels, makes tracking more difficult [23]. However, apps still engage in tracking using other methods, such as fingerprinting, whereby individual users are identified probabilistically [23]. Balash et al. conducted a large-scale longitudinal analysis of the App Store [4]. Over the course of 36 weeks, the researchers analyzed weekly snapshots of 1.6 million apps and their associated privacy labels [4]. They found that only 60.5% of apps provided privacy labels, and that 42.1% of apps with labels indicated they did not collect any data. However, the researchers concluded that since many apps that reported no data collection were likely to do so, the privacy labels were likely inaccurate [4]. Xiao et al. conducted a small scale privacy analysis by comparing the privacy labels and binaries of 5,102 iOS apps, finding many instances of non-compliance [42].

2.4 Natural Language Processing Techniques for Document Classification

Natural Language Processing (NLP) has been a cornerstone in semantically understanding and parsing privacy policies. Automated compliance systems, such as MAPS and ATLAS, require the use of NLP to analyze hundreds of thousands of privacy policies that would otherwise take human annotators years to accomplish. Past research has detailed the effectiveness of using NLP techniques to analyze privacy policies. In particular, Story et al. explores how using established techniques for processing privacy policies – such a TF-IDF vectorization and logistic regression – can achieve state-of-the-art results when used for compliance analysis [38]. The authors formulate the identification of privacy practice statements as a classification problem using an ensemble of classifiers (i.e. one classifier per privacy practice) [38]. Specifically, the authors focus on assigning annotation labels to policy segments (which roughly correspond to paragraphs) – with labeled data being pulled from the APP-350 corpus [38].

This paper replicates the approach, with several key differences. First, instead of focusing on policy segments, this work formulates identification of data collection as a document classification problem. Second, instead of using annotator labeled data (such as from the APP-350 corpus), we use developer reported iOS privacy labels to describe the text of privacy policies. Unlike annotator labeled data, privacy labels may not be consistent with the text of privacy policies because developers struggle to accurately create them [15] [24]. Finally, we experiment with additional state-of-the-art model architectures for document classification. Similar to the paper,

we use an ensemble-based approach where each data type is identified using its own classifier.

Document classification is an extensively researched field. Early research in the field yielded model architectures utilizing convolutional layers such as KimCNN and Char-CNN [21] [45]. Later work utilized recurrent neural networks (RNNs) with attention mechanisms, such as Hierarchial Attention Networks, which were able to achieve state-of-the-art results [43]. In extreme multi-label document classification scenarios, where each document can be assigned a set of labels numbering in the hundreds or thousands, XML-CNN has been shown to produce state-of-the-art results [25]. However, since the search space for this paper is relatively small (32 labels), we use a simpler approach: training individual models for each label.

Current state-of-the-art techniques in NLP are largely based on the transformer model, which sidesteps traditional methods such as recurrence [40]. Instead it relies entirely on using attention mechanisms to learn global dependencies between inputs and outputs [40]. Transformer based models, such as BERT and RoBERTa have been shown to achieve incredible results in NLP [11] [26]. In the field of document classification, a tailored version of BERT, dubbed DocBERT, has been shown to achieve state-of-the-art results on popular document classification datasets [1]. However, documents are typically longer than the maximum token length allowed by BERT, RoBERTa, and DocBERT: 512 tokens, where each token roughly corresponds to one word. To alleviate this, the Longformer model was proposed. Longformer increased the maximum token length to eight-fold what previous models were capable of [5]. This is particularly useful, as the median length of privacy policies – 2500 words – is quite large [28]. In an application of transformer based models to privacy, Ravichander et al. fine tuned BERT to answer privacy related questions about mobile app privacy policies [32].

While transformer based models are able to achieve incredible results, they come with tremendous computational overhead. Recognizing this, Adhikari et al. proposed the RegLSTM, a regularized LSTM that was able to outperform several transformer models in popular document classification tasks [2]. Importantly, RegLSTM requires fewer resources to run, resulting in a far lower overhead than transformer based models [2]. Moreover, the reduced overhead enables RegLSTM to process documents far larger than even the Longformer model. In our experimentation, we were able to process privacy policies in excess of 9000 words. More information about the formulation of our ensemble-based classifier will be presented in Chapter 4.

Chapter 3

A Distributed Pipeline for Automated Analysis

3.1 iOS App Sampling Strategy

We first began by identifying a candidate list of apps to analyze. Prior work has relied on crawling mobile app stores for app discovery; however, we found that not to be necessary for this work [49]. Fortunately, Apple publishes a categorized, alphabetical list of all available iOS Apps, including popular apps per category [17]. On January 29th, 2023, we systematically crawled and scraped the website to assemble a list of 918,293 unique apps available on the United States iOS App Store. Of those apps, 4,846 are classified as popular apps.

Next, we devised a sampling strategy to pick a subset of apps to analyze. Conducting a simple random sample of the entire App Store is the easiest way to generate a representative sample of the entire App Store; however, this approach is likely to miss heavy-hitters: frequently downloaded apps that are more likely to be present on users's devices – popular apps. Conversely, sampling only popular apps leads to a biased representation of the App Store, as many apps are missed. We devised a hybrid sampling strategy to create a set of apps likely to be present on user's devices *and* an unbiased representation of apps available on the iOS App Store: we sampled all 4,846 popular apps in addition to a randomly selected set of 350,000 non-popular apps. In total, our final dataset comprised of 354,725 apps, as some app listings were unable to be loaded.

3.2 Identifying Privacy Policies

iOS apps are required to provide a URL to their privacy policy. However, in many cases, these URLs would lead to landing pages, or other unrelated webpages. To accurately obtain privacy policies, we developed a logistic regression classier to determine if pages were English-language privacy policies, similar to prior work [49]. We collected 918 webpages linked to by iOS app privacy policies, of which 618 (67.3%) where legitimate policies, and 300 were unrelated webpages. Beautiful Soup 4 (BS4) was used to extract text from the downloaded pages, which were then vectorized using scikit-learn's TF-IDF implementation. We used k-fold cross validation to grid search over several regularization values (`C = [1.0, 2.0, 512, 1024]`).

We then evaluated our trained classifier on a separate unseen test set, with 64 positive examples and 42 negative examples. Our classifier was able to achieve 98.1% accuracy with an F1 score of 98.4% (precision = 100%, recall = 96.9%).

To run our system within a reasonable amount of time, we only classified pages directly linked to by the provided privacy policy URL. We did not crawl websites to discover privacy policies.

3.3 Design of a Distributed Infrastructure

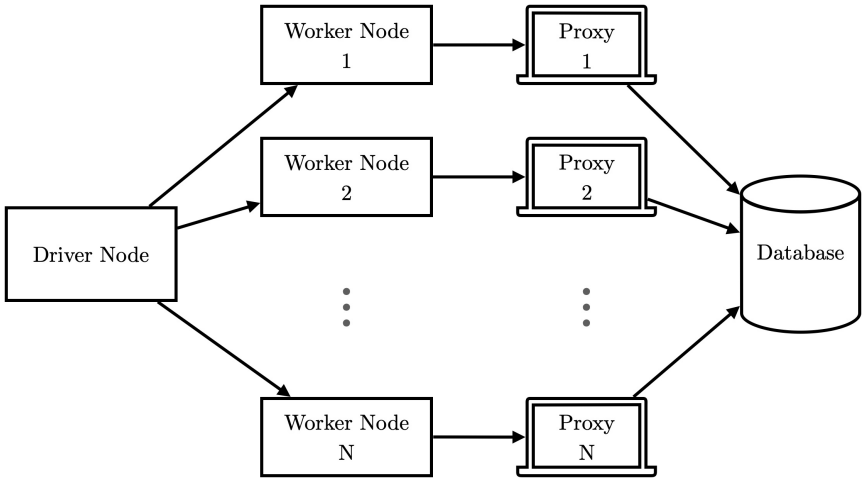


Figure 3.1: A diagram of the ATLAS data collection pipeline.

The scale of our study necessitated the design, development, and deployment of a highly parallelizable and distributed data collection pipeline. This enabled us to collect all data within a relatively short window. To this end, we created an infrastructure as depicted in Figure 3.1. We utilized a driver node to coordinate work between N worker nodes. Each worker node ran a headless Firefox browser to replicate a real-world browser. This gave us the ability to capture dynamically loaded content and follow any webpage redirects – emulating the experience of a real user. Since we continuously hit the same base domain (`https://apps.apple.com`) to scrape app pages, rate-limiting was a concern. To mitigate that, we utilized a pool of N proxy servers (one per worker) to increase the number of available IP addresses. The SOCKS5 protocol was used to connect worker nodes to proxies. After webpages were retrieved, they were saved in a shared database.

We began data collection on January 29$^{\text{th}}$, 2023, and our system ran until January 31$^{\text{st}}$, 2023. We completed data collection in two phases. Phase 1 focused on downloading app listings from the iOS App Store, and Phase 2 focused on downloading privacy policies. We used slightly different configurations for each phase. Phase 1 utilized one driver node with 49 worker nodes and 49 proxies; whereas Phase 2 used one driver node with 80 worker nodes and no proxies. Proxies were not required in Phase 2 since privacy policies were hosted on different domains, so rate-limiting was not a concern.

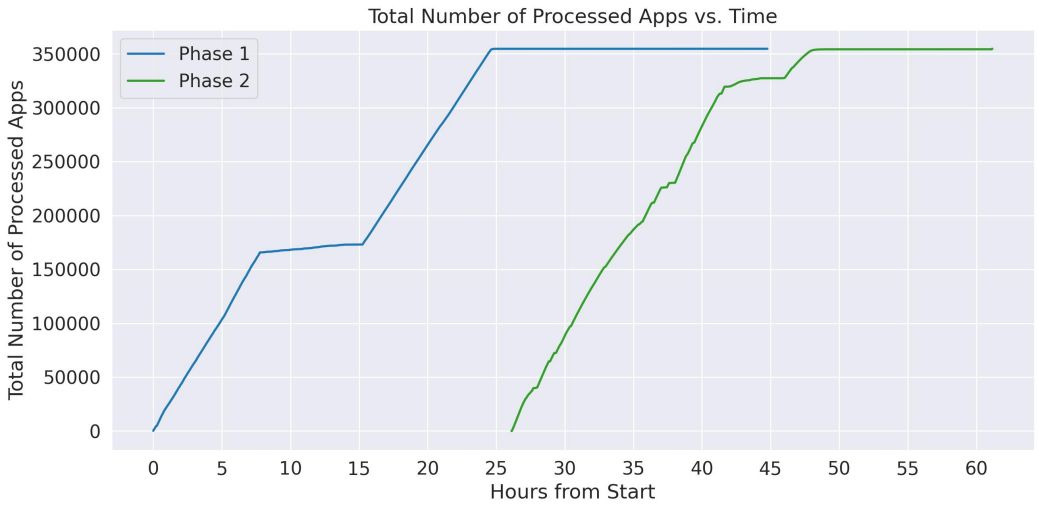


Figure 3.2: A diagram of the number of processed apps over time. Data collection began on January 29th, 2023 and ended on January 31st, 2023.

As depicted in Figure 3.2, Phase 1 ran at a rate of approximately 21,000 apps per hour. Around the 7.5 hour mark, our proxy servers initiated a nightly-reboot cycle, which caused the rate to diminish. After manual intervention, our system picked back up around the 15 hour mark. 9 proxy servers were no longer responsive, so our rate slowed to approximately 19,000 apps per hour. Phase 2 began around hour 26 at an average rate of 20,500 apps per hour.

At around hour 45, we reran Phases 1 and 2 in an attempt to re-download app listings and privacy policies that timed out. We reran Phase 2 once more around the 60 hour mark to collect the final set of privacy policies. In total, we were able to collect 345,725 apps.

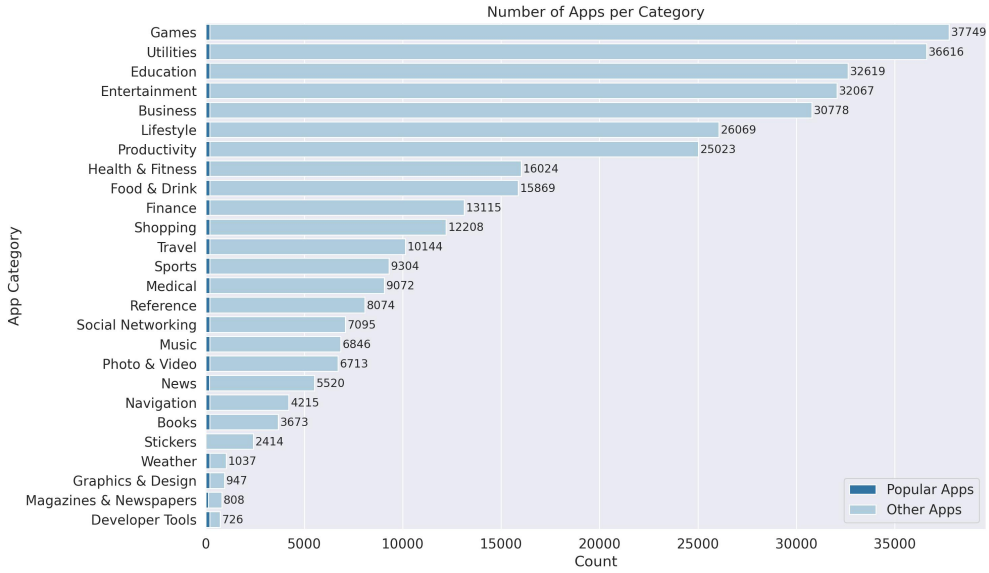
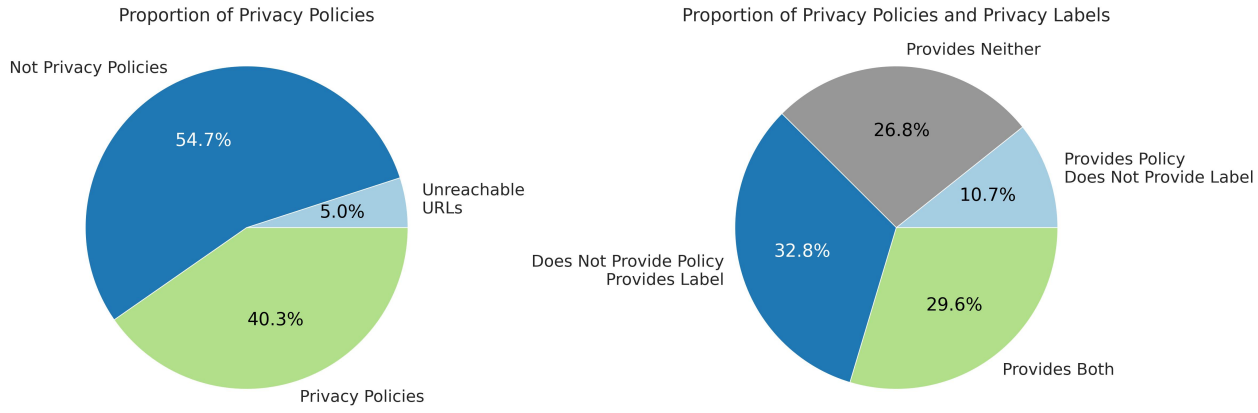


Figure 3.3: A depiction of the number of apps per category.

Through the use of our system, we were able to collect a total of 345,725 apps. This number was slightly lower than the total number of sampled apps, as some apps were unreachable. We then conducted a preliminary analysis of downloaded data. As depicted in Figure 3.3, Games were the most popular app within our sample and Developer Tools was the least popular. Additionally, popular apps were approximately uniformly distributed among all categories; however, the Stickers category was an exception, with no popular apps.

3.4 Privacy Policy and Privacy Label Adoption



(a) Proportion of Privacy Policies (b) Proportion of Privacy Policies and Privacy Labels

Figure 3.4: Figure 3.4a depicts the proportion of webpages linked to by reported privacy policy URLs in the iOS App Store. Figure 3.4b depicts the proportion of apps providing privacy policies and privacy labels in the iOS App Store.

We found several interesting trends upon analysis of downloaded apps. Even though apps are required to link to privacy policies, a substantial number of apps provided extraneous links instead. As depicted in Figure 3.4b, 5.0% of apps provided dead links. Moreover, 54.7% of apps provided links that led to extraneous webpages, such as landing pages, home pages, and 404s. Only 40.3% of apps provided direct links to legitimate privacy policies, which we characterized as accessible privacy policies.

Next, we analyzed the adoption rate of privacy labels. Apple began requiring apps published or updated after December 2020 include privacy labels. Interestingly, we discovered that 62.5% of sampled apps provided privacy labels. A number substantially higher than those that provided accessible privacy policies. As depicted in Figure 3.4b, only 29.6% of apps (105,131 apps) provided both accessible privacy policies and privacy labels. We focused on this subset of apps to conduct our compliance analysis.

Finally, we analyzed the most common types of reported data collection, as show in Figure 3.5. Unsurprisingly, the most common data type collected was Crash Data, followed by Product Interaction and Email Address. Interestingly, Gameplay Content was the second least reported type of data collected, even though Games were the most common type of app in our dataset

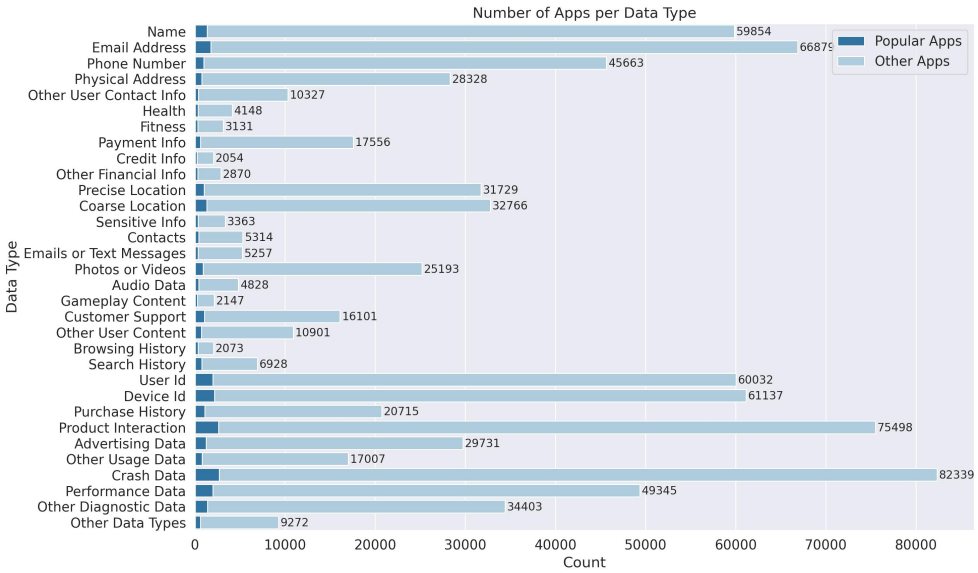


Figure 3.5: A depiction of the number of apps using declaring collection of each data type.

(and in the App Store), as shown in Figure 3.3. Also indicated is the number of popular apps reporting collection of each data type: the distribution of collection across data types for popular apps approximately follows the same distribution as apps available in the broader App Store.

Chapter 4

Autogenerating Privacy Labels from Privacy Policies

We next focused our efforts on predicting privacy labels from privacy policies. We formulated the task as a supervised multi-class, multi-label document classification problem, where the input is a privacy policy, and the output is a set of collected data types, as indicated by the privacy policy. We targeted the same 32 data types included in iOS privacy labels (2.1.2).

iOS privacy labels include what data types are collected, and require developers to indicate how each data type is used (up to six purposes), if it is linked to the user, and if it can be used to track the user. Instead, we focused only on identifying which data types were collected. This enabled us to more precisely focus efforts by reducing search space complexity by a factor of eight.

To avoid challenges associated with single model multi-label classification, we treated predicting each data type as an independent binary classification problem. That is, we created 32 models, where each model would be responsible for predicting a single data type. While this did increase overall runtime, it greatly simplified model complexity.

4.1 Dataset Construction

Unlike prior research, which used high quality annotator labeled privacy policies, we relied solely on developer reported privacy labels to learn from privacy policies [38] [49]. We began by filtering our downloaded set of apps to those which provided both privacy policies and privacy labels. This left us with 105,131 iOS apps. However, upon analysis, we found many privacy policies to be shared among apps – developers likely reused the same privacy policy among many of their apps. We determined policy uniqueness by comparing privacy policy URLs for exact matches, though a more sophisticated system that looks at the cosine similarity of privacy policy text would likely find matches among identical policies with different URLs. However, we found our system to be adequate, as its simplicity yielded performant runtime. In total, we found 34.5% of policies to be duplicates, leaving us with 68,863 unique privacy policies.

Care was taken to preserve the structure of differing privacy labels for identical privacy policies. We assumed that shared privacy policies are written generally, so as to be applicable to

multiple apps. Privacy labels, however, are constructed on a per app basis; therefore, they only represent a subset of the privacy policy. As a result, we reasoned that when duplicate policies were merged together, their privacy labels needed to be combined by taking the union of collected data types. For example, if the same privacy policy was associated with one privacy label that declared collection of {`Name, Email Address, Precise Location`} while another declared {`Name, Physical Address`}, the resulting merged privacy label would declare {`Name, Email Address, Physical Address, Precise Location`}.

4.2 Sampling Procedure

While using developer reported privacy labels enabled us to build a training corpus almost a hundred times larger than previous work, our data was expected to be somewhat "noisy" [38] [49]. Smaller scale analyses in the past suggest that privacy labels may not be accurate [24] [15]. While much work has been dedicated to learning from noisy, mislabeled data ([37], [46]), it is "unfair and unreasonable to have noise in the" testing data [6]. We outline a technique for importance sampling to help reduce noise when creating training and testing datasets. For clarity, we define the following terms used within this section:

- A **Positive** instance is a privacy policy associated with a privacy label that reported a particular data type as being collected.
- A **Negative** instance is a privacy policy associated with a privacy label that reported a particular data type as not being collected.

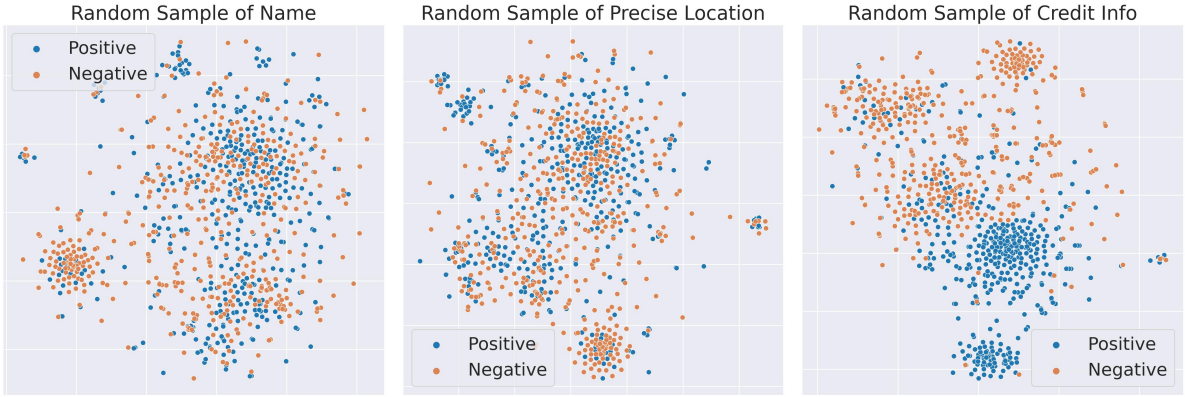


Figure 4.1: Random samples of Name (left), Precise Location (middle), and Credit Info (right). Each point is a t-SNE representation of a privacy policy TF-IDF embedding. Point colors correspond to developer reported values for data collection where "Positive" indicates the data type was collected and "Negative" indicates the particular data type was not collected.

Figure 4.1 characterizes how noisy the underlying data was. We provided random samples for three data types: Name, Precise Location, and Credit Info. For each data type, we randomly sampled 500 policies with positive instances, and 500 policies with negative instances. We then extracted an embedding from each policy using TF-IDF vectorization and created a two-dimensional representation using t-SNE [39]. For the Name and Precise Location data types,

there is significant overlap between positive and negative instances and no clear distinction between the two groups; this suggests that many policies might be mislabeled. Credit Info, on the other hand, has a much clearer distinction between the two groups, likely because accurately disclosing collection and processing of financial information is mandated by law [8].

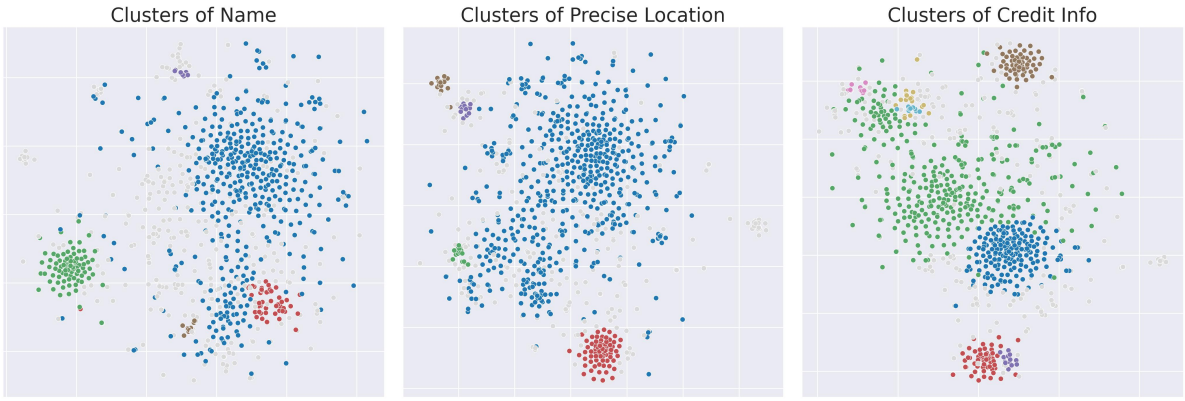


Figure 4.2: Clusters appearing in random samples of Name (left), Precise Location (middle), and Credit Info (right). Each point is a t-SNE representation of a privacy policy TF-IDF embedding. Point colors correspond to DBSCAN clusters, where light grey points are outliers, and all other colored points are cluster inliers.

However, as depicted in Figure 4.2, natural clusters tended to appear within the randomly sampled data. Points were clustered using DBSCAN – since the number of clusters was not known a priori and the underlying data contains some noise – after performing Latent Semantic Analysis (LSA) on the high dimensional TF-IDF embeddings [14] [12]. LSA was performed by using TruncatedSVD to project the high-dimensional data onto 10 components. We reasoned that these clusters are semantically similar privacy policies, and should therefore all have the same privacy label (either positive or negative). For example, if a cluster had 20 positive instances and 5 negative instances, then the negative instances could potentially be mislabeled, since the content of their privacy policies were similar to many positive instances.

We assigned clusters a positive or negative label by conducting a two-proportion z-test that compared the incidence of positive to negative labels within a cluster. If a cluster had a statistically significantly larger proportion (i.e. $p < 0.05$) of one class than the other, we assigned the cluster the label of the more prominent class. In cases where determining cluster labels was inconclusive (i.e. $p \geq 0.05$), we disregarded the cluster entirely. We then computed centroids for all positive and negative clusters.

Finally, to construct a dataset of size N, with $\frac{N}{2}$ positive and $\frac{N}{2}$ negative examples, we conducted a random oversample of N instances per class. Then, for each class, we selected the $\frac{N}{2}$ examples closest to the class centroids, effectively sampling points closer to centroids with higher probability than those farther away. We also equally weighted the contribution of each class centroid. Each of the C centroids had approximately $\frac{N}{2C}$ examples associated with it. To account for edge cases where no clusters were identified, we fell back to a simple random sample.

This process was used to construct the test ($N = 150$), validation ($N = 150$), and training sets ($N = 1000$), for each data type. We had 4,376 unique examples within the test sets, 4,262

unique examples within the validation sets, and 22,262 unique examples within the training sets. In total, we used 30,900 unique privacy policies to train and evaluate our classifiers.

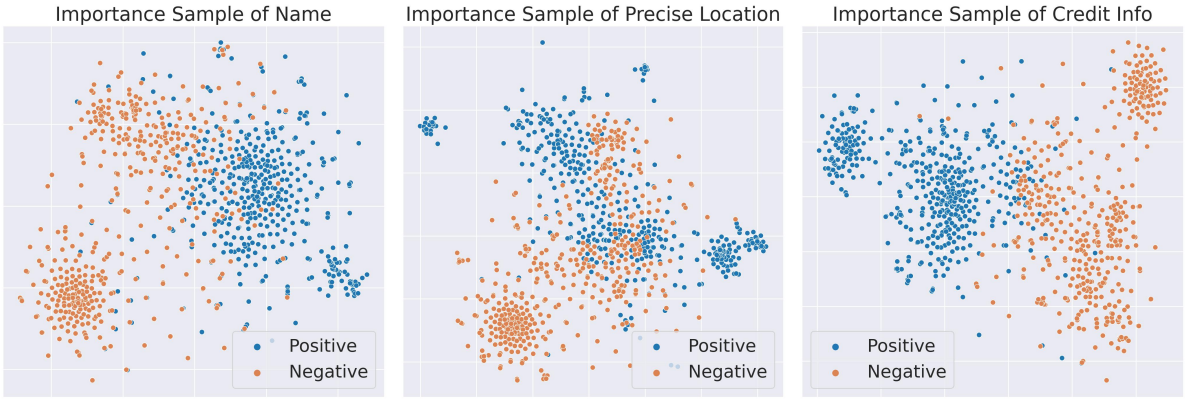


Figure 4.3: Importance samples of Name (left), Precise Location (middle), and Credit Info (right). Points are visualized using the same methodology as in Figure 4.1.

Intuitively, our sampling procedure can be thought of focusing on high-density areas with large amounts of information more likely to be labeled correctly. Points on the edge have a lower probability of being selected: they are assumed to be less likely to be labeled correctly, therefore contributing less useful information. Figure 4.3 demonstrates how importance sampling creates a clear separation between the positive and negative classes, in contrast to random sampling shown in Figure 4.1.

4.3 Model Selection

After constructing the datasets, we trained using several model architectures. We used logistic regression as our baseline architecture, with a similar configuration to prior work [38]. We then graduated to using more complex architectures: multilayer perceptron (MLP), RegLSTM, BERT, RoBERTa, and Longformer. We conducted extensive hyperparameter tuning for each model, per data type. Since we were training a separate model per data type, we created a final "ensemble" model utilizing a combination of architectures that maximized validation Macro F1 Score.

Models were trained using PyTorch on four NVIDIA GeForce RTX 2080 Ti GPUs [30]. In total, it took approximately 48 hours to hyperparameter tune and train 32 models across 6 different architectures.

4.3.1 Logistic Regression (Baseline)

We first began with logistic regression, our baseline model. We vectorized privacy policies using scikit-learn's TF-IDF vectorizer using English stop words `stop_words='english'` with unigrams and bigrams (`ngram_range=(1,2)`), similar to prior work [38] [49]. We hyperparameter tuned by grid searching across batch sizes of 50 and 100; weight decay of 0 and 10^{-5}; and TF-IDF binary term counts (`binary=True` and `binary=False`). Hyperparameter tuning was performed per data type. After finding optimal configurations, we trained for 20 epochs per data type.

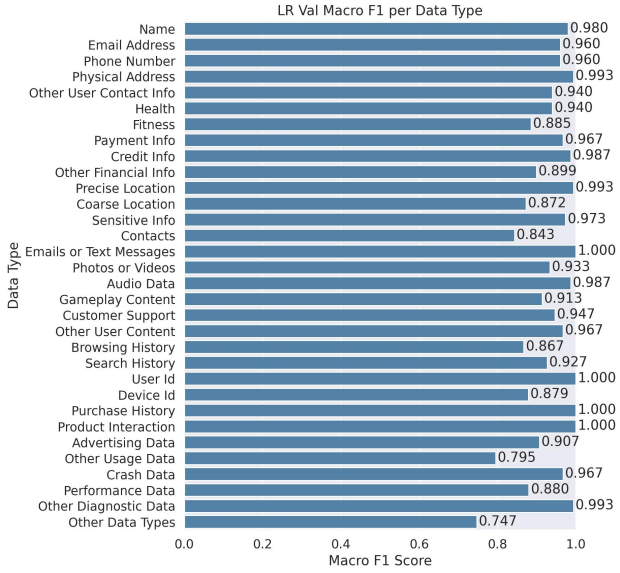


Figure 4.4: Logistic Regression (baseline) validation results per data type.

As shown in Figure 4.4, logistic regression served as a strong baseline, with an average Macro F1 score across data types of 93.4%. In fact, several data types – Emails or Text Messages, User ID, Purchase History, and Product Interaction – had Macro F1 scores of 100%.

4.3.2 Multilayer Perceptron

We next trained using a slightly more complex architecture: a multilayer perceptron (MLP) with a single hidden layer. Privacy policy vectorization was identical to the baseline model (4.3.1). However, we performed a more extensive hyperparameter tuning step by grid searching across batch sizs of 50 and 100; weight decay of 0 and 10^{-5}; TF-IDF binary term counts (`binary=True` and `binary=False`); hidden layer sizes of 128 and 256; and dropout probabilites of 0.4 and 0.6. Similar to the baseline model, hyperparameter tuning was performed per data type. After finding optimal configurations, we trained for 20 epochs per data type.

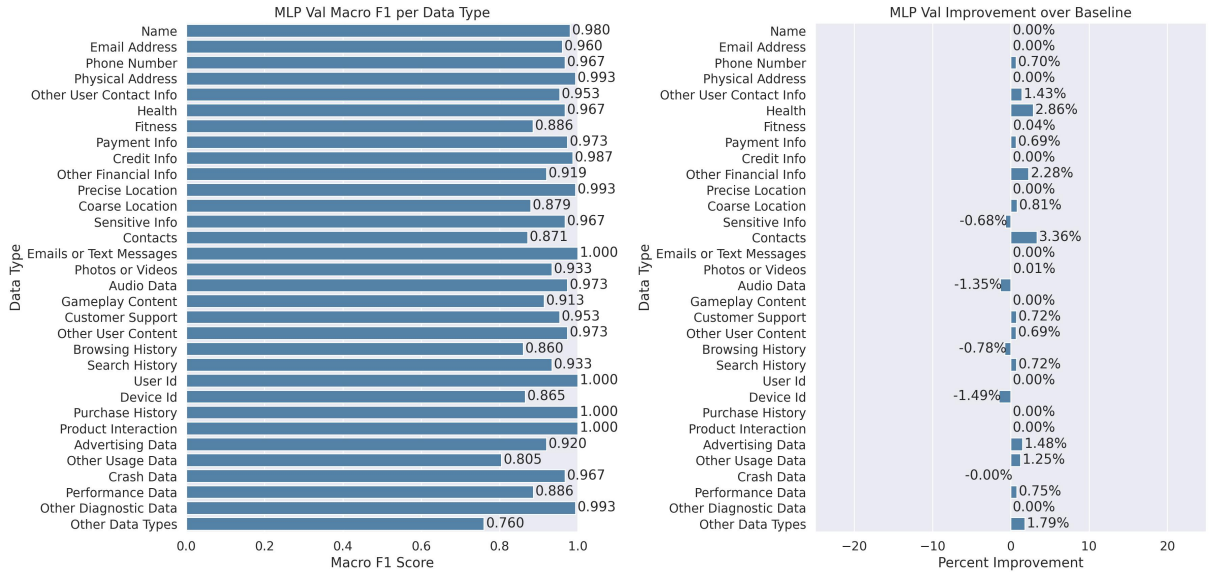


Figure 4.5: MLP validation results per data type are shown on the left, and improvement over the baseline model is shown on the right.

As shown in Figure 4.5, using an MLP introduced modest performance improvements (measured by percentage improvement over baseline Macro F1 scores) across several data types, with the largest increases being among Health, Other Financial Info, and Contacts.

4.3.3 RegLSTM

Next, we evaluated the RegLSTM model. Introduced by Adhikari et al., the RegLSTM falls into the category of recurrent neural networks [2]. Importantly, RegLSTM has lower computational overhead than more complex models (such as BERT), yet is still able to achieve state-of-the-art results on popular benchmark datasets [2]. We use the implementation of RegLSTM provided by the Hedwig toolkit [3]. We made one change from the provided implementation: instead of using `word2vec` embeddings, we utilized `GloVe`, finding that the latter embedding method to have broader support within PyTorch [29] [31]. While the maximum sequence length for RegLSTM is unbounded, we chose to use 5,000 words. This struck a balance between runtime and privacy policy length (the median policy length was close to 2,000 words).

We then performed hyperparameter tuning using grid search across 1 and 2 network layers; `static` and `non-static` embed modes; and weight decay of 0, 10^{-5}, and 10^{-4}. Similar to prior architectures, hyperparameter tuning was performed per data type. After finding optimal configurations, we trained for 20 epochs per data type.

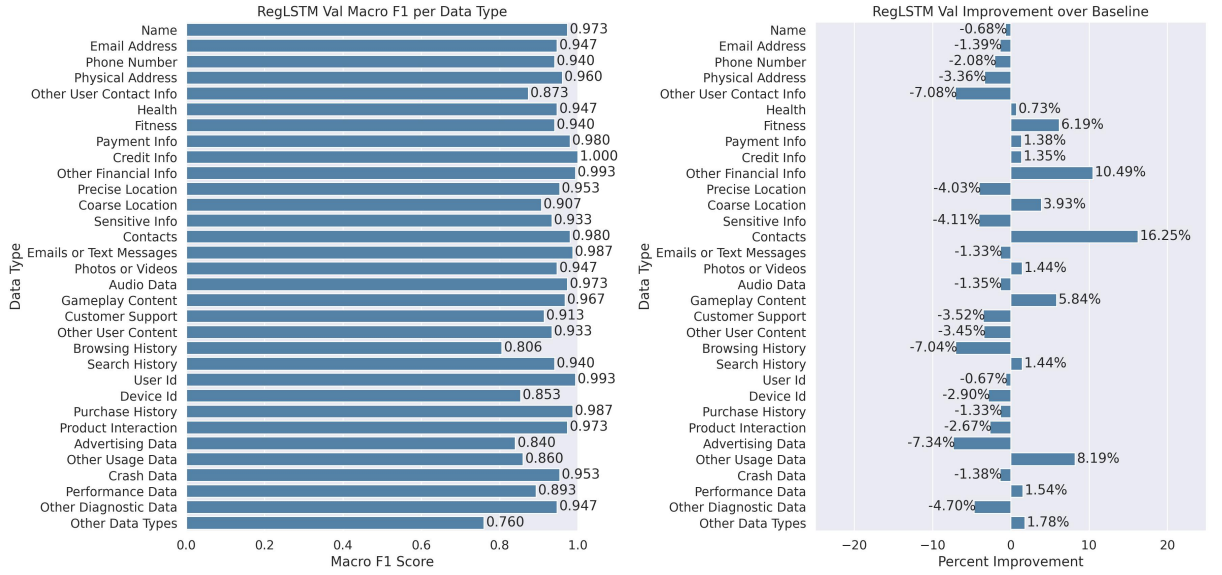


Figure 4.6: RegLSTM validation results per data type are shown on the left, and improvement over the baseline model is shown on the right.

As shown in Figure 4.6, the RegLSTM shows substantial improvement in several categories, such as Fitness, Other Financial Info, Contacts, and Other Usage Data. However, several categories, such as Other User Contact Info and Advertising Data, perform notably worse than baseline.

4.3.4 BERT

We next move onto transformer based models. We fine-tuned BERT for sequence classification, using the implementation provided by HuggingFace and the `bert-base-uncased` base model [41]. BERT has been shown to achieve state-of-the-art results on popular NLP benchmark datasets, and we hope to replicate similar performance in the context of this work [11]. As before, we perform hyperparameter tuning by grid searching across weight decays of 10^{-5}, 2×10^{-4}, and 10^{-3}. Hyperparameter tuning was performed per data type, and after finding optimal configurations, we trained for 10 epochs.

Since BERT has a maximum token length of 512, we truncated privacy policies to the first 512 tokens.

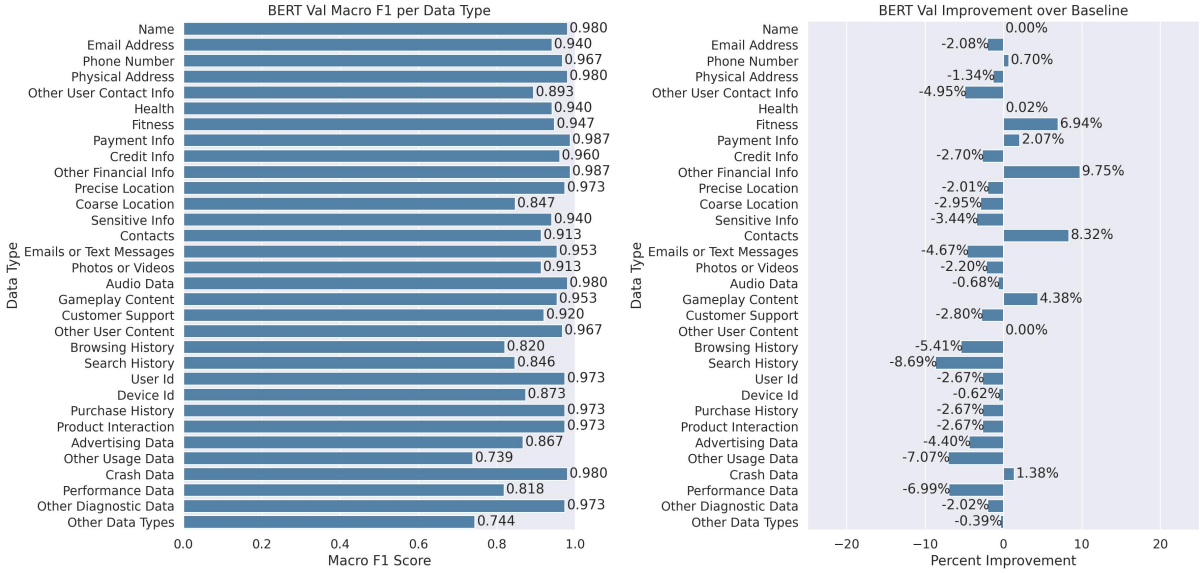


Figure 4.7: BERT validation results per data type are shown on the left, and improvement over the baseline model is shown on the right.

Figure 4.7 shows that BERT perfoms significantly better than baseline in several categories, including Fitness, Payment Information, and Audio Data. However, like the RegLSTM, it also performs notably worse on several data types, such as Search History.

4.3.5 RoBERTa

We also evaluated RoBERTa on our dataset. RoBERTa is robustly optimized version of BERT that outperforms it on standard NLP benchmarks [26]. We a fine-tuned RoBERTa for sequence classification using the implementation provided by HuggingFace and the `roberta-base` base model [7]. As before, we perform hyperparameter tuning by grid searching across weight decays of 10^{-5}, 2×10^{-4}, and 10^{-3}. Hyperparameter tuning was performed per data type, and after finding optimal configurations, we trained for 10 epochs.

Since RoBERTa has a maximum token length of 512, we truncated privacy policies to the first 512 tokens.

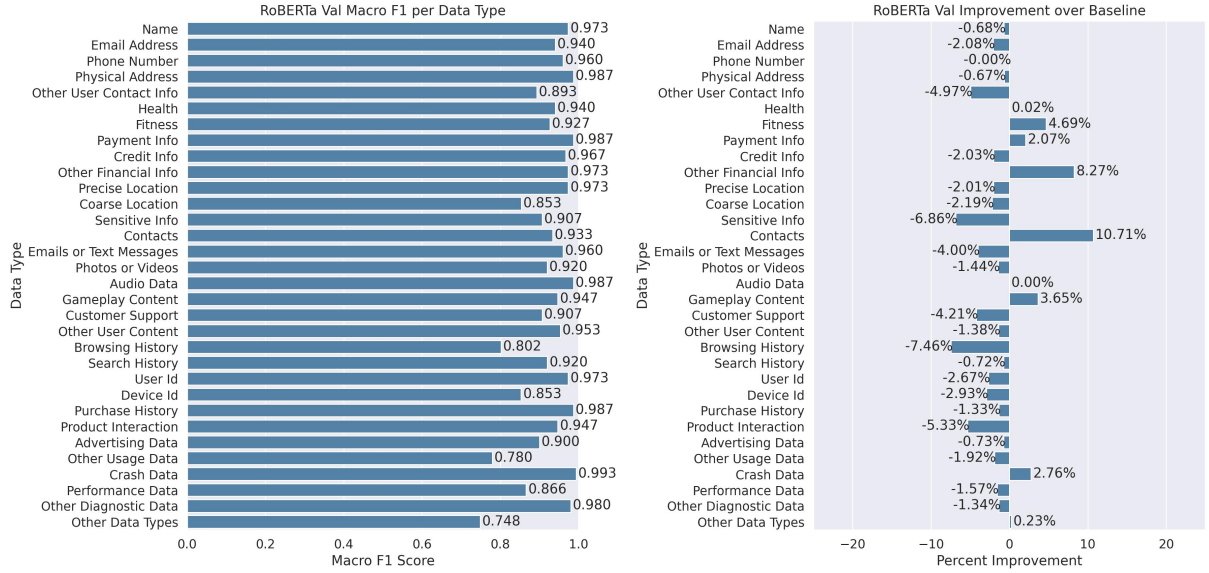


Figure 4.8: RoBERTa validation results per data type are shown on the left, and improvement over the baseline model is shown on the right.

As shown in Figure 4.8, RoBERTa outperforms the baseline model in several categories, such as Contacts and Crash Data. However, similar to BERT, it underperforms in several other categories, such as Search History.

4.3.6 Longformer

Finally, we evaluated Longformer on our dataset. While BERT and RoBERTa are able to outperform the baseline model for several data types, they are limited by their token length of 512 tokens. Since a token approximately corresponds to a single word (although typically it is a subword), BERT and RoBERTa exclude large chunks of privacy policies, since the median length is approximately 2,000 words. Longformer aims to solve this by expanding the maximum token length to 4096. However, due to computational constraints (i.e. exceeding GPU memory constraings), we had to limit token length to 1536 (triple that of BERT / RoBERTa). As before, we perform hyperparameter tuning by grid searching across weight decays of 10^{-5}, 2×10^{-4}, and 10^{-3}. Hyperparameter tuning was performed per data type, and after finding optimal configurations, we trained for 8 epochs.

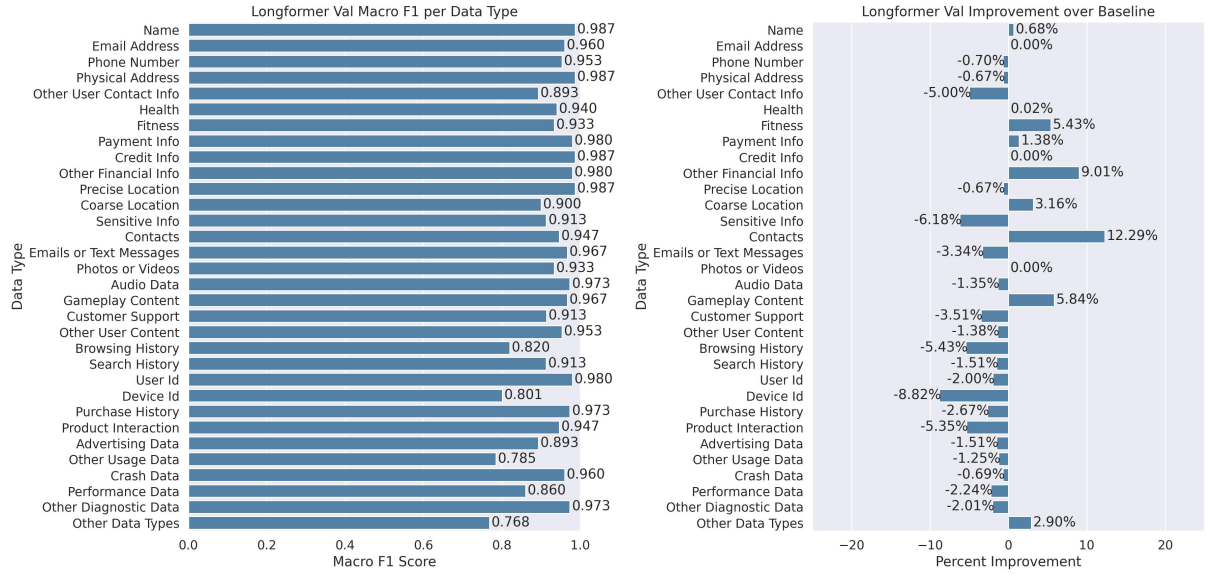


Figure 4.9: Longformer validation results per data type are shown on the left, and improvement over the baseline model is shown on the right.

Figure 4.9 shows that Longformer was able to outperform the baseline model considerably in several categories. Of note, it was the only model to outperform baseline for the Name and Other Data Type categories. However, like previous transformer models, it underperformed for several data types, such as Device ID.

4.3.7 Ensemble

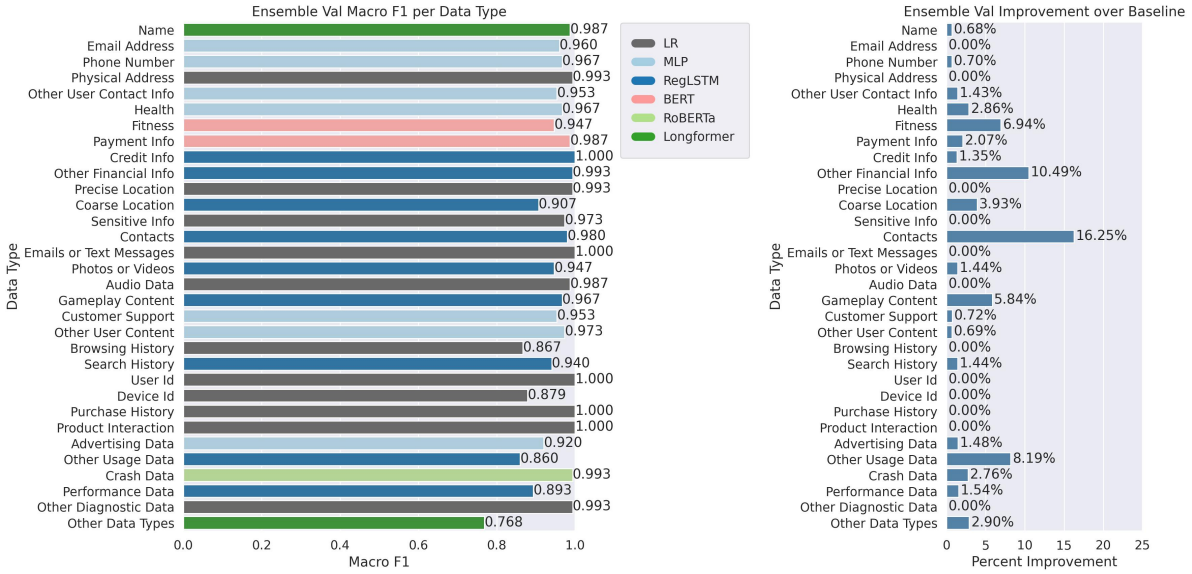


Figure 4.10: Ensemble validation results per data type are shown on the left, and improvement over the baseline model is shown on the right.

Our final model was constructed with an ensemble of architectures. Since each data type was trained separately, we selected the architecture that maximized the Macro F1 score for that data type. In the case that multiple models have identical Macro F1 scores, we selected the simplest model. Figure 4.10 shows the selection of architecture per data type, as well as overall improvement over baseline. No singular data model dominated; however, the baseline model was highly competitive compared to more complex models.

4.4 Model Performance

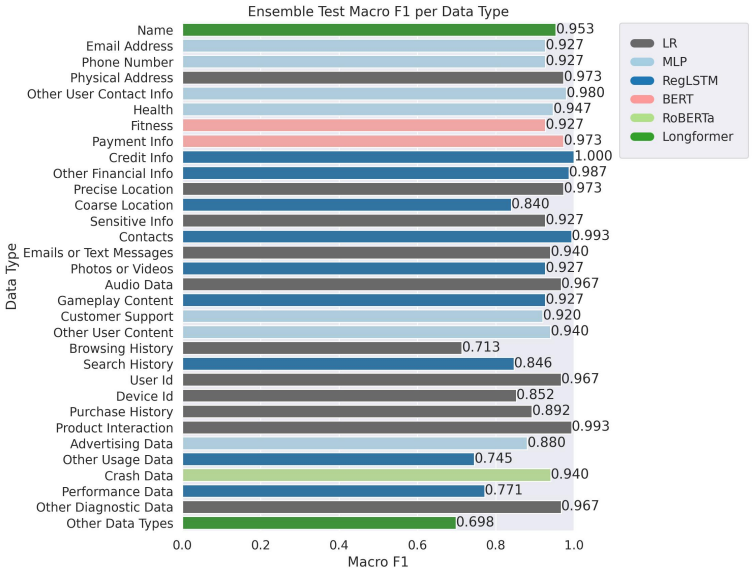


Figure 4.11: Ensemble test results per data type

Finally, we evaluated our ensemble of models on an unseen set of test data. Figure 4.11 shows the test Macro F1 score achieved per data type. Overall, we were able to achieve an average accuracy of 91.3% and an average Macro F1 score of 91.3% across all classes. Notably, we were able to achieve a Macro F1 score of 100% for Credit Info, with several other data types being the high-90s.

Table 4.1 provides detailed statistics about model performance on the test dataset.

Table 4.1: The following table summarizes the performance of the final ensemble model on the test dataset.

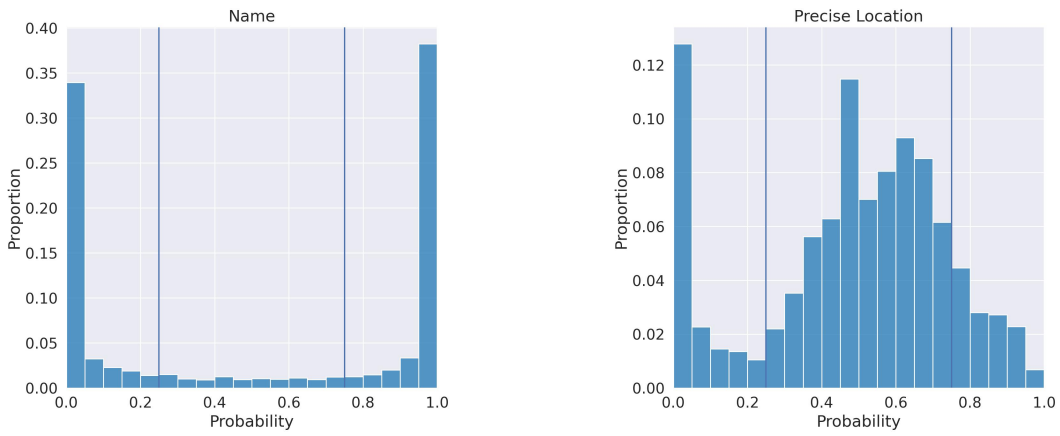
	Test Acc	Acc	Macro F1	F1	Prec	Recall
Name	0.95	[0.99, 0.92]	0.95	[0.95, 0.95]	[0.93, 0.99]	[0.99, 0.92]
Email Address	0.93	[0.89, 0.96]	0.93	[0.92, 0.93]	[0.96, 0.90]	[0.89, 0.96]
Phone Number	0.93	[0.91, 0.95]	0.93	[0.93, 0.93]	[0.94, 0.91]	[0.91, 0.95]
Physical Address	0.97	[0.95, 1.00]	0.97	[0.97, 0.97]	[1.00, 0.95]	[0.95, 1.00]
Other User Contact Info	0.98	[0.96, 1.00]	0.98	[0.98, 0.98]	[1.00, 0.96]	[0.96, 1.00]
Health	0.95	[0.92, 0.97]	0.95	[0.95, 0.95]	[0.97, 0.92]	[0.92, 0.97]
Fitness	0.93	[0.93, 0.92]	0.93	[0.93, 0.93]	[0.92, 0.93]	[0.93, 0.92]
Payment Info	0.97	[0.99, 0.96]	0.97	[0.97, 0.97]	[0.96, 0.99]	[0.99, 0.96]
Credit Info	1.00	[1.00, 1.00]	1.00	[1.00, 1.00]	[1.00, 1.00]	[1.00, 1.00]
Other Financial Info	0.99	[0.97, 1.00]	0.99	[0.99, 0.99]	[1.00, 0.97]	[0.97, 1.00]
Precise Location	0.97	[0.96, 0.99]	0.97	[0.97, 0.97]	[0.99, 0.96]	[0.96, 0.99]
Coarse Location	0.84	[0.79, 0.89]	0.84	[0.83, 0.85]	[0.88, 0.81]	[0.79, 0.89]
Sensitive Info	0.93	[0.89, 0.96]	0.93	[0.92, 0.93]	[0.96, 0.90]	[0.89, 0.96]
Contacts	0.99	[1.00, 0.99]	0.99	[0.99, 0.99]	[0.99, 1.00]	[1.00, 0.99]
Emails or Text Messages	0.94	[0.88, 1.00]	0.94	[0.94, 0.94]	[1.00, 0.89]	[0.88, 1.00]
Photos or Videos	0.93	[0.91, 0.95]	0.93	[0.93, 0.93]	[0.94, 0.91]	[0.91, 0.95]
Audio Data	0.97	[0.96, 0.97]	0.97	[0.97, 0.97]	[0.97, 0.96]	[0.96, 0.97]
Gameplay Content	0.93	[0.92, 0.93]	0.93	[0.93, 0.93]	[0.93, 0.92]	[0.92, 0.93]
Customer Support	0.92	[0.92, 0.92]	0.92	[0.92, 0.92]	[0.92, 0.92]	[0.92, 0.92]
Other User Content	0.94	[0.89, 0.99]	0.94	[0.94, 0.94]	[0.99, 0.90]	[0.89, 0.99]
Browsing History	0.71	[0.73, 0.69]	0.71	[0.72, 0.71]	[0.71, 0.72]	[0.73, 0.69]
Search History	0.85	[0.80, 0.89]	0.85	[0.84, 0.85]	[0.88, 0.82]	[0.80, 0.89]
User Id	0.97	[0.96, 0.97]	0.97	[0.97, 0.97]	[0.97, 0.96]	[0.96, 0.97]
Device Id	0.85	[0.77, 0.93]	0.85	[0.84, 0.86]	[0.92, 0.80]	[0.77, 0.93]
Purchase History	0.89	[0.80, 0.99]	0.89	[0.88, 0.90]	[0.98, 0.83]	[0.80, 0.99]
Product Interaction	0.99	[1.00, 0.99]	0.99	[0.99, 0.99]	[0.99, 1.00]	[1.00, 0.99]
Advertising Data	0.88	[0.88, 0.88]	0.88	[0.88, 0.88]	[0.88, 0.88]	[0.88, 0.88]
Other Usage Data	0.75	[0.67, 0.83]	0.75	[0.72, 0.77]	[0.79, 0.71]	[0.67, 0.83]
Crash Data	0.94	[0.92, 0.96]	0.94	[0.94, 0.94]	[0.96, 0.92]	[0.92, 0.96]
Performance Data	0.77	[0.68, 0.87]	0.77	[0.75, 0.79]	[0.84, 0.73]	[0.68, 0.87]
Other Diagnostic Data	0.97	[0.93, 1.00]	0.97	[0.97, 0.97]	[1.00, 0.94]	[0.93, 1.00]
Other Data Types	0.70	[0.77, 0.63]	0.70	[0.72, 0.68]	[0.67, 0.73]	[0.77, 0.63]

Chapter 5

Compliance Analysis

After training our ensemble-based classifier, we used it to predict privacy labels for the remaining privacy policies. After removing training data, we were left with a set of privacy policies corresponding to 61,596 iOS apps. The following analysis is presented for those apps.

5.1 Characterizing Potential Compliance Issues



(a) Probability Distribution for Name

(b) Probability Distribution for Precise Location

Figure 5.1: Shown here are two probability distributions (Name and Precise Location) after running our ensemble-based classifier on the entire set of privacy policies (excluding training data). This figure characterizes classifier confidence for each data type.

We first begin by characterizing a potential compliance issue. For an arbitrary app, let P be the set of data types collected as disclosed by an app's privacy policy, and let L be the set of data types collected as disclosed by its privacy label. Let D represent the set of all data types, and let $d \in D$ be a single data type. We use $\hat{d}$ to denote a predicted data type. For our purposes, $|D| = 32$. By definition, we can write $P \subseteq D$ and $L \subseteq D$.

The first type of potential compliance issue is an **Incomplete Privacy Policy** (often shortened to Incomplete Policy in the rest of the section):

$$\exists\, d \in D, \text{ such that } (\hat{d} \notin P) \wedge (d \in L)$$

Intuitively, an incomplete policy means that a privacy policy does not disclose the collection of a data type, while its developer reported privacy label does. While, this type of discrepancy can be interpreted as an over-reported privacy label disclosure, privacy labels in iOS default to non-disclosure. That is, developers have to manually indicate their app collects a particular data type, which is indicative of an underlying reason for disclosure. As a result, we believe that characterizing this type of discrepancy as an "Incomplete Privacy Policy" is more reasonable.

The second type of potential compliance issue is an **Incomplete Privacy Label** (often shortened to Incomplete Label):

$$\exists\, d \in D, \text{ such that } (\hat{d} \in P) \wedge (d \notin L)$$

Intuitively, an incomplete label is when collection of a data type is disclosed within a privacy policy, but not within the developer reported privacy label. Similar to before, this type of discrepancy can be interpreted as an over-reported privacy policy; however, we reason that the effort to include legal text about data collection within a policy is indicative of an underlying reason. Therefore, we believe it to be more appropriate to characterize this type of discrepancy as an "Incomplete Privacy Label."

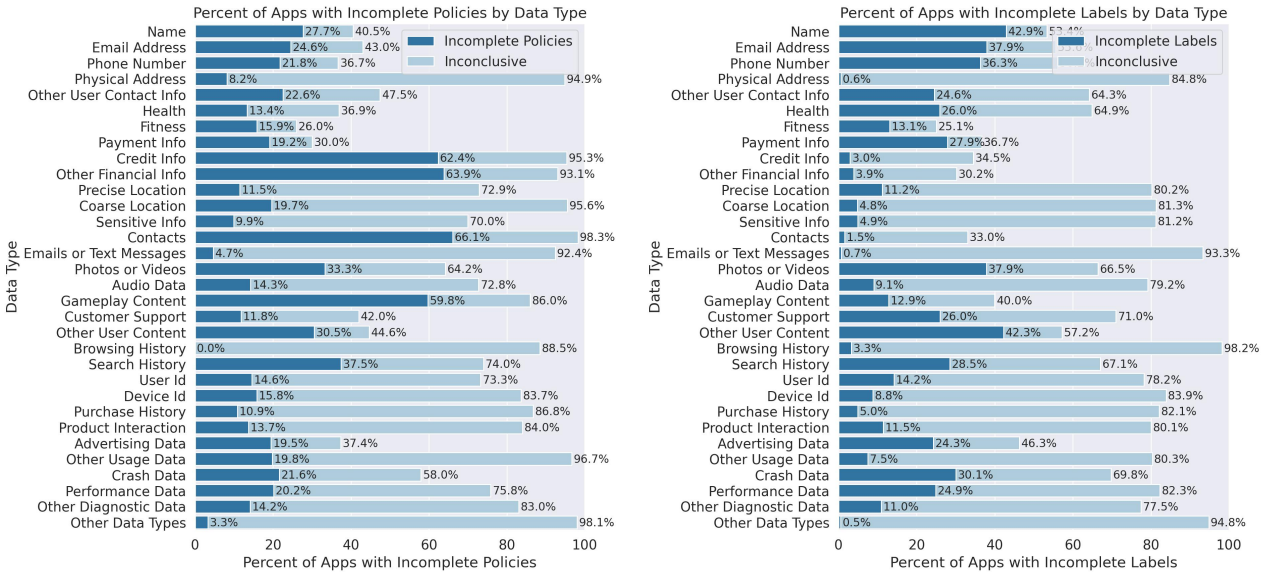
Potential compliance issues are on a per data type basis. This means that the total number of potential compliance issues per app is capped at 32 (one per data type).

We also took care to ensure that we only identified potential compliance issues with high probability. Figure 5.1, shows the probability that a policy collects a particular data type (i.e. $p_{\hat{d}\in P}$). Since we formulated this as a binary classification problem for each data type, $p_{\hat{d}\notin P} = 1 - p_{\hat{d}\in P}$. For example, as shown in 5.1a, most policies have a high probability of collecting name, or a high probability of not collecting Name; however, for many policies, it is uncertain that they collect Precise Location (Figure 5.1b), as many probabilities are near 50%. So, we only considered

$$(\hat{d} \notin P) \iff (p_{\hat{d}\in P} < 0.25)$$
$$(\hat{d} \in P) \iff (p_{\hat{d}\in P} > 0.75)$$

which is depicted by the two vertical lines in Figures 5.1a and 5.1b. Intuitively, we only considered predictions with high probability; otherwise, we classify the prediction as "Inconclusive."

5.2 Potential Compliance Issues by Data Type



(a) Incomplete Policies

(b) Incomplete Labels

Figure 5.2: Potential compliance issues by data type.

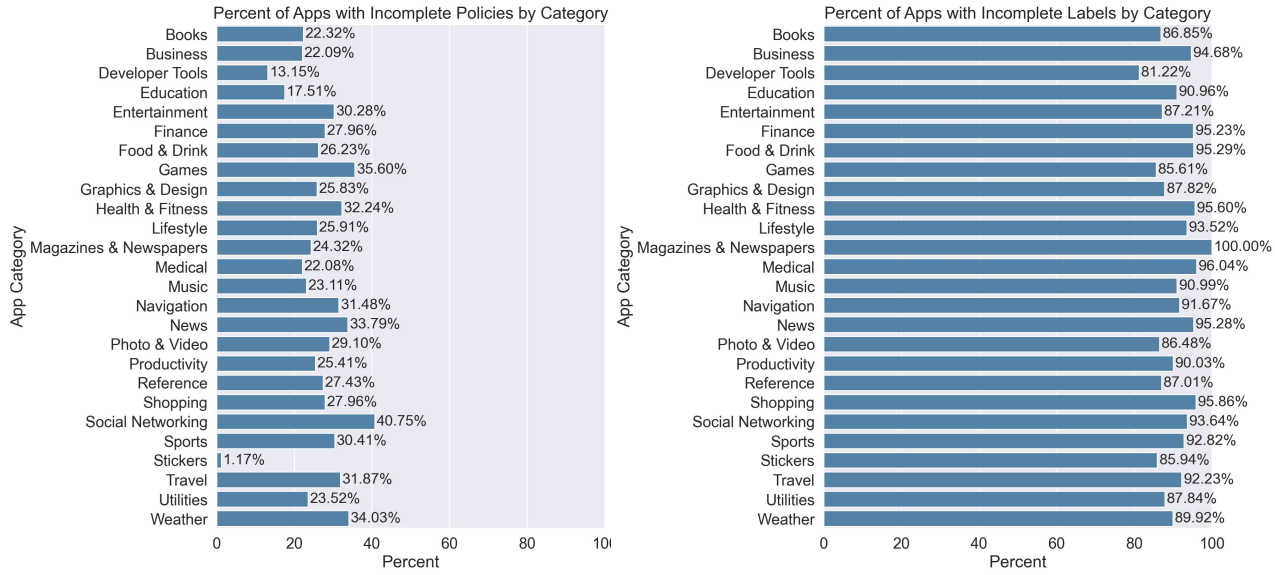
We offer a breakdown of compliance issues by data type, as show in Figure 5.2. Figure 5.2a, shows the rate of incomplete policies by data type. The percentage for each data type, $d \in D$, is the total number of incomplete policies for d divided by the total number of apps where $L \Rightarrow d_1$. For example, 27.7% of apps have privacy policies that do not declare collection of Name, even when their privacy labels do.

Similarly, Figure 5.2b shows the rate of incomplete labels by data type. The percentage for each data type, $d \in D$, is the total number of incomplete labels for d divided by the total number of apps where $L \Rightarrow d_0$. For example, 42.9% of apps have privacy labels that do not declare collection of Name, even when their privacy policies do.

Of particular note are the high rates of incomplete policies within financial disclosures: 62.4% of apps declaring Credit Info and 63.9% of apps declaring Other Financial Info on their privacy labels have incomplete privacy policies. This particular set of potential compliance issues could be a violation of the Gramm-Leach-Bliley Act [8].

While it may seem that incomplete policies are more prevalent than incomplete labels, it is important to note that for most data types, d, the number of apps declaring they do not collect d is typically higher than the number of apps declaring they do collect d. We discuss this more in Section 5.4.

5.3 Potential Compliance Issues by Category



(a) Incomplete Policies

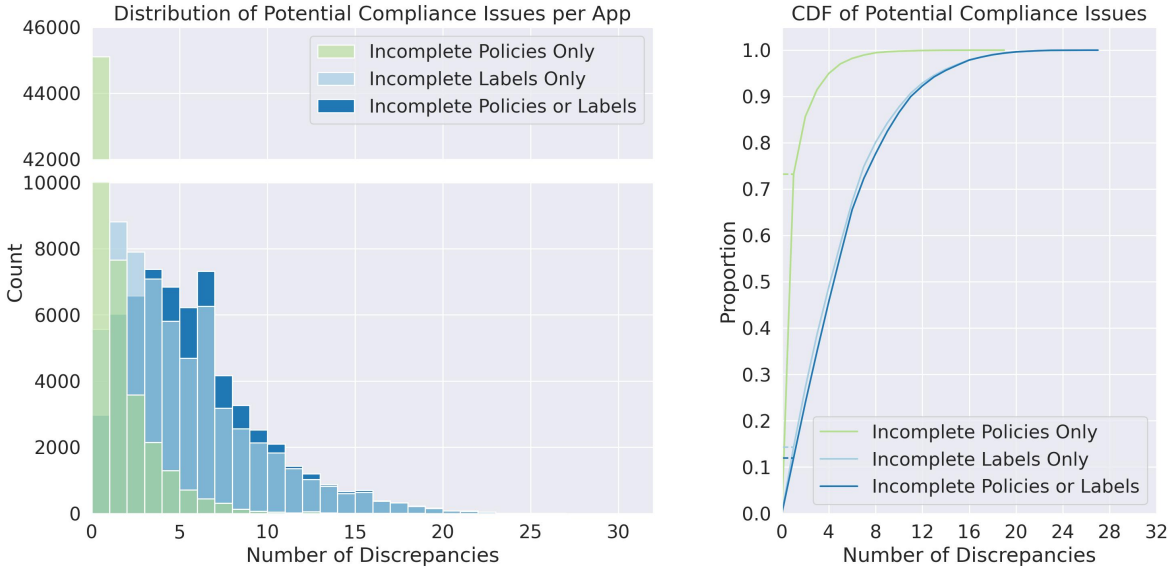
(b) Incomplete Labels

Figure 5.3: Potential compliance issues by category.

Next, we characterize the rates of potential compliances by app category, as shown in Figure 5.3. Figure 5.3a depicts the percentage of apps per category with at least one incomplete policy discrepancy. Figure 5.3b similarly depicts the percentage of apps per category with at least one incomplete label discrepancy.

As is shown within the two graphs, both types of potential compliance issues are approximately uniformly distributed across categories, with the notable exception of the Stickers category, when analyzing incomplete privacy policies (likely due to many errors being inconclusive). Interestingly, the rates between the two types of errors are quite different: apps are far less likely to have incomplete policies (the maximum discrepancy rate per category is 40.75%) than incomplete labels (where all categories have discrepancy rates between 80% and 100%).

5.4 Distribution of Potential Compliance Issues



(a) Distribution of Potential Compliance Issues

(b) CDF of Potential Compliance Issues

Figure 5.4: On the left is a graph of the distribution of the different types of potential compliance issues, and on the right is a CDF of the different types of potential compliance issues.

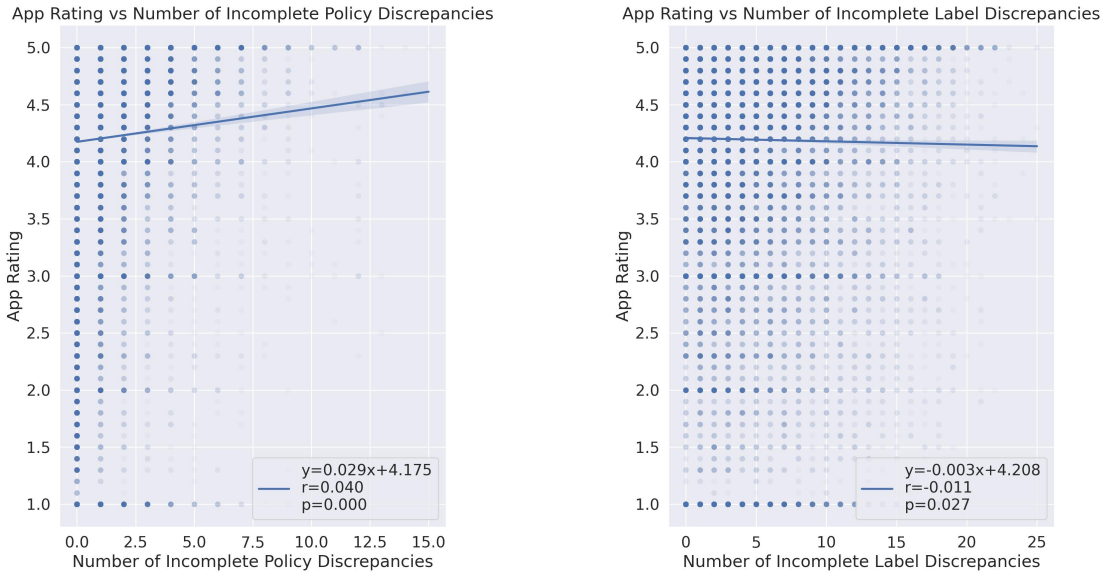
Table 5.1: This table provides a cumulative distribution of potential compliance issues.

	Incomplete Policies (%)	**Incomplete Labels (%)**	**Both (%)**
1 or more discrepancies	26.8	85.7	88.0
2 or more discrepancies	14.3	72.9	76.1
3 or more discrepancies	8.5	61.3	65.0

Figure 5.4 shows the distribution and cumulative distribution of potential compliance issues. In particular, incomplete policies are far less common than incomplete labels, with apps having an average of 0.62 incomplete policy discrepancies and 4.698 incomplete label discrepancies. When looking at the combination of incomplete policy and label errors, apps have 5.32 potential compliance issues on average. The lower rate of incomplete policy errors is clearly represented by Figure 5.4a, as the bucket for 0 compliance issues is far larger than for incomplete labels (note that there is a break in the graph between 10,000 and 42,000 on the y-axis, so the depiction is less exaggerated).

Figure 5.4b depicts the CDF for each data type, which makes it much clearer to examine differences between distributions. Of note, 26.8% of apps have at least one incomplete policy discrepancy and 85.7% of apps have at least one incomplete label discrepancy. When analyzing both incomplete policies and labels, 88.0% of apps have at least one discrepancy. We provide additional percentages in Table 5.1.

5.5 App Rating vs Number of Potential Compliance Issues



(a) Rating vs Incomplete Policy Discrepancies (b) Rating vs Incomplete Label Discrepancies

Figure 5.5: The graph on the left depicts the correlation with respect to incomplete policy discrepancies, and the graph on the right depicts the correlation with respect to incomplete label discrepancies. Darker points represent a larger number of apps (iOS App Store ratings are discretized in tenths and discrepancies are integers). The shaded region around each line represents a 95% confidence interval.

We also compared the trend between app rating and number of potential compliance issues, as shown in Figure 5.5. Surprisingly, for incomplete policy compliance issues, we found a weak, significant *positive* correlation ($r = 0.04$, $p < 0.05$) between app rating and number of incomplete policy discrepancies. That is, as the number of incomplete policy discrepancies increase, the app rating tends to *increase*.

While the previous result was surprising, the relationship between app rating and incomplete label discrepancies is as expected: we found a weak, significant negative correlation ($r = -0.011$, $p < 0.05$). That is, as the number of incomplete label discrepancies increase, the app rating tends to decrease.

5.6 Potential Compliance Issues Among Popular vs Other Apps

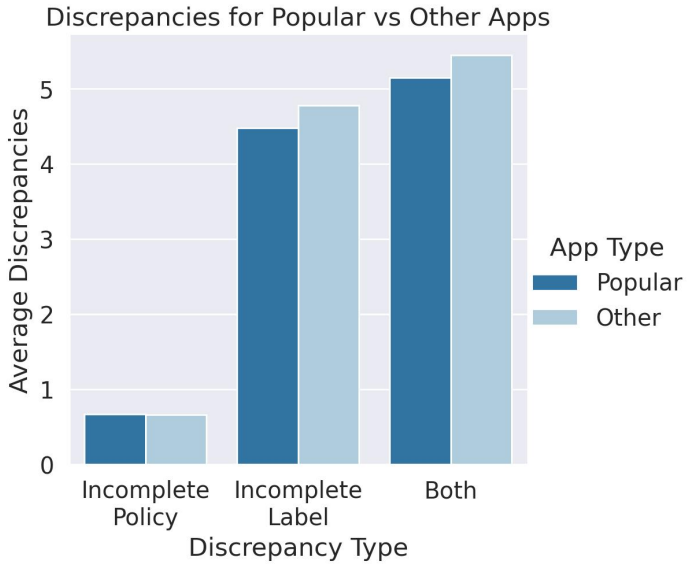


Figure 5.6: A depiction of the average number of discrepancies among popular and other apps. Discrepancies types are categorized by Incomplete Policies, Incomplete Labels, and Both.

Table 5.2: This table provides the average number of discrepancies among popular and other apps, categories by discrepancy type. Discrepancy types with significant difference ($p < 0.05$) are signified with *

	Avg Discrepancies for Popular Apps	**Avg Discrepancies for Other Apps**	**p value**
Incomplete Policies	0.67	0.66	0.877
Incomplete Labels	4.48	4.78	0.005*
Both	5.15	5.44	0.004*

Finally, we compared the incidence of discrepancies between popular and other apps. As shown in Figure 5.6 and Table 5.2, other apps (i.e. unpopular apps), tend to have a statistically significantly higher number of incomplete label discrepancies, on average, than popular apps ($p < 0.05$). Consequently, when looking across all discrepancies, other apps also have a higher number of discrepancies, on average, than popular apps ($p < 0.05$). There is no significant difference between the incidence of incomplete policy discrepancies between popular and other apps ($p > 0.05$).

We conclude from this comparison that popular apps, on average, have fewer discrepancies than their counterparts. As popular apps represent apps likely to be on user's devices, this trend is reassuring; however, the incidence of discrepancies among popular apps is still high, at an average of 5.15 discrepancies per app.

Chapter 6

Discussion

In Section 3, we outlined the development of a pipeline to systematically download and analyze iOS App Store listings. We found 62.5% of apps to provide privacy labels. This is approximately the same, albeit slightly higher than reported by Balash et al. in their work [4]. The slightly higher percentage in our study likely reflects our study being conducted several months after Balash et al.'s, allowing more time for apps to include privacy labels.

We also introduced a mechanism to identify if a webpage linked to by a privacy policy URL was a legitimate policy. For efficiency reasons, we chose not to perform a limited crawl in cases where the webpage was a landing page, and required further navigation to the privacy policy. While this likely reduced the reported number of apps with privacy policies, we believe that our approach mimics the actions of the average consumer, who is unlikely to navigate complex webpages to find a privacy policy.

In Chapter 4, we outlined a technique for predicting iOS privacy labels from the text of privacy policies. To our knowledge, this is the first work that has attempted such a task. However, it is not without limitations. For one, we relied on noisy data to train our classifiers. To mitigate this we used an importance sampling approach to reduce variance within the data. However, this may have had the effect of biasing the training data towards similar policies, lowering effective recall in cases where outlier policies were labeled correctly, but were not included due to lack of similarity.

Moreover, in certain cases, Apple stipulates that privacy labels may optionally disclose certain data collection, but are not required to do so [16]. This could lead to results that overestimate potential discrepancies when looking at optional privacy label disclosures. In principle, because less data collection is typically seen as desirable, developers would typically be expected to only disclose collection when required to do so. So, assuming that our training data is biased towards required disclosures, our resulting classifiers could be expected to indicate positive instances only if they are truly required.

In Chapter 5, we described a large-scale analysis of apps available on the iOS App Store. Prior work had already provided evidence that iOS privacy labels can be inaccurate [24], [44]. Our study adds to this evidence by comparing disclosures made in privacy policies to those in privacy labels. To the extent that discrepancies are indicative of potential compliance issues, we find that as many as 88% of apps have at least one potential compliance issue, with apps having an average of 5.32 potential compliance issues. These results appear generally consistent with

prior work, albeit in slightly different contexts. For example, Zimmeck et al. report a mean of 2.89 potential compliance issues per app in their large-scale analysis of Android apps; however, they characterized potential compliance issues as discrepancies between the text of privacy policies and static code analysis of Android apps [49]. Our work, on the other hand, compares disclosures within privacy policies to those in privacy labels. We also find that Incomplete Label discrepancies were more common that Incomplete Policy discrepancies. This could likely be a byproduct of privacy policies being written to be more general and permissive than privacy labels. In particular, some policies are known to apply to multiple apps, whereas all privacy labels are specific to a given app. In addition, privacy labels may often be created by developers and may be authored with the intent to only disclose those practices an app actually engages in. In contrast, privacy policies are known to often be "written by lawyers, for lawyers" and, as a result, can be expected to be more general.

Additionally, Balash et al. found in a review of 1.6 million apps available on the App Store that 42.1% of apps with labels indicate they do not collect any data, which they claim is unlikely to be true. Our work provides concrete evidence for their claim by showing that incomplete privacy labels are the primary source of errors, which is consistent with their results [4].

Chapter 7

Future Work

Several avenues exist to continue this work in the future. We offer three potential areas of exploration that naturally extend this work: a detailed privacy policy prevalence analysis, predicting more detailed privacy labels, and combining our analysis with static and dynamic code analysis.

7.1 A Detailed Privacy Policy Analysis

First, we could explore the prevalence of privacy policies within the iOS App Store in more detail. For example, we currently only determine if the webpage linked to by the privacy policy URL provided by apps is a legitimate privacy policy; however, in some cases, the URL leads to a privacy landing page, and the actual privacy policy can be found by clicking a link somewhere on that page. Future work could examine the "depth" of these privacy policies – how many pages from the root do users need to navigate to arrive at a privacy policy?

7.2 Predicting Detailed Privacy Labels

Another avenue for exploration involves predicting more detailed privacy labels from privacy policies. We currently only predict if a privacy policy declares collection of a particular data type. However, iOS privacy labels go further, providing information on how collected data types are used, if they are linked to the user, and if they can be used to track the user. Future work could investigate if it is possible to accurately determine how collected data is used from the text of privacy policies. One approach could be to formulate privacy label prediction as a two-step process. First, future work could use our classifiers to determine if a privacy policy collected a particular data type. Then, they could implement an additional set of classifiers to determine how the data type was used.

Moreover, providing textual evidence to corroborate predicted data types could also be investigated. For example, users might have questions about how data is being used beyond the capabilities of privacy labels, such as "How long does this app retain my data?" Prior work has focused on answering these types of questions [32]. Future work could combine [32] with our classifiers to produce more detailed summaries of privacy policies.

7.3 Static and Dynamic Code Analysis

Finally, future research could combine this work with static and dynamic code analysis. These types of mechanisms enables analyses that provide concrete evidence to ground claims about data collection practices by analyzing app code and app behavior upon execution. This evidence can then be compared to the text of privacy policies to determine potential compliance issues more concretely. For example, Privacy Label Wiz is able to conduct static analyses on iOS apps to determine certain types of data collection [15]. Future work could combine our techniques – determining data collection within privacy policies – to determine if the text of privacy policies matches the code of iOS apps.

7.4 Conclusion

Privacy labels have been proposed as usable mechanisms to help users better understand salient data practices found within privacy policies. In December 2020, Apple began requiring that all iOS apps include privacy labels, arguably the largest adoption of privacy labels to date; however, prior work has questioned the accuracy of such labels [24] [15].

In this work, we introduced the Automated Privacy Label Analysis System (ATLAS). ATLAS enabled us to provide a detailed analysis of 354,725 iOS apps available on the United States App Store. We found that privacy policy accessibility and privacy label adoption is relatively low, with only 62.5% of apps providing privacy labels. We also developed an ensemble-based classier that was able to accurately predict privacy labels from privacy policies with 91.3% accuracy. We then used our classifier to conduct a compliance analysis, finding several interesting trends. For example, 88% of apps had at least one discrepancy between the text of their privacy policy and their privacy label. On average, we found iOS apps to have 5.32 discrepancies. These discrepancies could potentially be indicative of compliance issues. We hope that our work enables a thorough review of privacy labels to help promote accurate privacy disclosures in the future.

Bibliography

[1] Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. DocBERT: BERT for document classification. *arXiv preprint arXiv:1904.08398*, 2019. 2.4

[2] Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. Rethinking complex neural network architectures for document classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4046–4051, 2019. 2.4, 4.3.3

[3] Data Systems Group at the University of Waterloo. Hedwig. `https://github.com/castorini/hedwig`, 2022. Accessed: 2023-03-13. 4.3.3

[4] David G Balash, Mir Masood Ali, Xiaoyuan Wu, Chris Kanich, and Adam J Aviv. Longitudinal analysis of privacy labels in the apple app store. *arXiv preprint arXiv:2206.02658*, 2022. 2.3, 6

[5] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020. 2.4

[6] Don Blaheta. Handling noisy training and testing data. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 111–116, 2002. 4.2

[7] Julien Chaumond. Roberta. `https://huggingface.co/docs/transformers/v4.26.1/en/model_doc/roberta`, 2023. Accessed: 2023-03-13. 4.3.5

[8] Federal Trade Commission. Gramm-leach-bliley act. `https://www.ftc.gov/business-guidance/privacy-security/gramm-leach-bliley-act`, 2023. Accessed: 2023-03-13. 4.2, 5.2

[9] Lorrie Faith Cranor. P3P: Making privacy policies more useful. *IEEE Security & Privacy*, 1(6):50–55, 2003. doi: 10.1109/MSECP.2003.1253568. 2.1.1

[10] Lorrie Faith Cranor, Serge Egelman, Steve Sheng, Aleecia M McDonald, and Abdur Chowdhury. P3p deployment on websites. *Electronic Commerce Research and Applications*, 7(3):274–293, 2008. 2.1.1

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2.4, 4.3.4

[12] Susan T Dumais et al. Latent semantic analysis. *Annu. Rev. Inf. Sci. Technol.*, 38(1):188–

230, 2004. 4.2

[13] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N Sheth. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):1–29, 2014. 2.3

[14] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996. 4.2

[15] Jack Gardner, Yuanyuan Feng, Kayla Reiman, Zhi Lin, Akshath Jain, and Norman Sadeh. Helping mobile application developers create accurate privacy labels. In *2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 212–230. IEEE, 2022. 2.1.3, 2.1.4, 2.4, 4.2, 7.3, 7.4

[16] Apple Inc. App privacy details - app store. URL `https://developer.apple.com/app-store/app-privacy-details/`. 2.1.2, 6

[17] Apple Inc. iTunes preview. `https://apps.apple.com/us/genre/ios-books/id6018`, 2023. Accessed: 2023-03-13. 3.1

[18] Patrick Gage Kelley, Joanna Bresee, Lorrie Faith Cranor, and Robert W. Reeder. A "nutrition label" for privacy. In *Proceedings of the 5th Symposium on Usable Privacy and Security*, SOUPS '09, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605587363. doi: 10.1145/1572532.1572538. URL `https://doi.org/10.1145/1572532.1572538`. 2.1.1

[19] Patrick Gage Kelley, Lucian Cesca, Joanna Bresee, and Lorrie Faith Cranor. Standardizing privacy notices: an online study of the nutrition label approach. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*, pages 1573–1582, 2010. 2.1.1

[20] Patrick Gage Kelley, Lorrie Faith Cranor, and Norman Sadeh. Privacy as part of the app decision-making process. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 3393–3402, 2013. 2.1.1, 2.1.2

[21] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1181. URL `https://aclanthology.org/D14-1181`. 2.4

[22] Simon Koch, Malte Wessels, Benjamin Altpeter, Madita Olvermann, and Martin Johns. Keeping privacy labels honest. *Proceedings on Privacy Enhancing Technologies*, 4:486–506, 2022. 2, 2.1.3

[23] Konrad Kollnig, Anastasia Shuba, Max Van Kleek, Reuben Binns, and Nigel Shadbolt. Goodbye tracking? impact of iOS app tracking transparency and privacy labels. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 508–520, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393522. doi: 10.1145/3531146.3533116. URL `https://doi.org/10.1145/3531146.`

3533116. 2.3

[24] Tianshi Li, Kayla Reiman, Yuvraj Agarwal, Lorrie Faith Cranor, and Jason I Hong. Understanding challenges for developers to create accurate privacy nutrition labels. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–24, 2022. 1, 2, 2.1.3, 2.1.4, 2.4, 4.2, 6, 7.4

[25] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pages 115–124, 2017. 2.4

[26] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. 2.4, 4.3.5

[27] Hunton & William LLP. Ten steps to develop a multilayered privacy notice. Technical report, The Center for Information Policy Leadership, 2007. 2.1.1

[28] Aleecia M McDonald and Lorrie Faith Cranor. The cost of reading privacy policies. *Isjlp*, 4:543, 2008. 1, 2.1.1, 2.4

[29] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. 4.3.3

[30] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. 2017. 4.3

[31] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162. 4.3.3

[32] Abhilasha Ravichander, Alan W Black, Shomir Wilson, Thomas Norton, and Norman Sadeh. Question answering for privacy policies: Combining computational and legal perspectives. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4949–4959, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1500. URL https://www.aclweb.org/anthology/D19-1500. 2.4, 7.2

[33] Joel Reardon, Álvaro Feal, Primal Wijesekera, Amit Elazari Bar On, Narseo Vallina-Rodriguez, and Serge Egelman. 50 ways to leak your data: An exploration of apps' circumvention of the android permissions system. In *28th USENIX security symposium (USENIX security 19)*, pages 603–620, 2019. 2.3

[34] Robert W Reeder, Patrick Gage Kelley, Aleecia M McDonald, and Lorrie Faith Cranor. A user study of the expandable grid applied to P3P privacy policy visualization. In *Proceedings of the 7th ACM workshop on Privacy in the electronic society*, pages 45–54, 2008. 2.1.1

[35] Joel R Reidenberg, Travis Breaux, Lorrie Faith Cranor, Brian French, Amanda Grannis, James T Graves, Fei Liu, Aleecia McDonald, Thomas B Norton, and Rohan Ramanath. Disagreeable privacy policies: Mismatches between meaning and users' understanding. *Berkeley Tech. LJ*, 30:39, 2015. 1

[36] Norman Sadeh, Alessandro Acquisti, Travis D Breaux, Lorrie Faith Cranor, Aleecia M McDonald, Joel R Reidenberg, Noah A Smith, Fei Liu, N Cameron Russell, Florian Schaub, et al. The usable privacy policy project. In *Technical report, Technical Report, CMU-ISR-13-119*. Carnegie Mellon University, 2013. 1

[37] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 4.2

[38] Peter Story, Sebastian Zimmeck, Abhilasha Ravichander, Daniel Smullen, Ziqi Wang, Joel Reidenberg, N Cameron Russell, and Norman Sadeh. Natural language processing for mobile app privacy compliance. In *AAAI Spring Symposium on Privacy-Enhancing Artificial Intelligence and Language Technologies*, 2019. 2.4, 4.1, 4.2, 4.3, 4.3.1

[39] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 4.2

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`. 2.4

[41] Thomas Wolf. Bert. `https://huggingface.co/docs/transformers/model_doc/bert`, 2023. Accessed: 2023-03-13. 4.3.4

[42] 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. 2.3

[43] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489, 2016. 2.4

[44] Shikun Zhang and Norman Sadeh. Do privacy labels answer users' privacy questions? 1, 6

[45] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015. 2.4

[46] Wenting Zhao and Carla Gomes. Evaluating multi-label classifiers with noisy labels. *arXiv preprint arXiv:2102.08427*, 2021. 4.2

[47] Sebastian Zimmeck, Ziqi Wang, Lieyong Zou, Roger Iyengar, Bin Liu, Florian Schaub, Shomir Wilson, Norman M Sadeh, Steven M Bellovin, and Joel R Reidenberg. Automated analysis of privacy requirements for mobile apps. In *NDSS*, 2017. 2.1.4, 2.3

[48] Sebastian Zimmeck, Peter Story, Rafael Goldstein, David Baraka, Shaoyan Li, Yuanyuan Feng, and Norman Sadeh. Compliance traceability: Privacy policies as software development artifacts. *Open Day for Privacy, Usability, and Transparency (PUT), Stockholm, Sweden*, 2019. 2.1.4

[49] Sebastian Zimmeck, Peter Story, Daniel Smullen, Abhilasha Ravichander, Ziqi Wang, Joel Reidenberg, N Cameron Russell, and Norman Sadeh. MAPS: Scaling privacy compliance analysis to a million apps. *Proceedings on Privacy Enhancing Technologies*, 2019(3):66–86, 2019. 2.3, 3.1, 3.2, 4.1, 4.2, 4.3.1, 6

[50] Sebastian Zimmeck, Rafael Goldstein, and David Baraka. PrivacyFlash Pro: Automating privacy policy generation for mobile apps. In *NDSS*, 2021. 2.1.4