Automatic Identification and Extraction of Privacy Policy Text: PoliScraper

Tong Jiao **Norman Sadeh**

April 2024
CMU-S3D-24-103

Software and Societal Systems
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This research has been partially supported by the National Science Foundation under its Security and Trustworthy Computing Program (grant CNS-1914486).

Keywords: Natural Language Processing, Privacy Policies

Abstract

Privacy policies are known to be long and challenging for people to read. To overcome this issue, a variety of natural language processing (NLP) techniques have been developed over the past dozen years to automate the analysis of these documents. The initial step in this process is the automatic extraction of privacy policies - typically from a website. While a number of tools have been proposed to extract privacy policy text, these tools all suffer from various limitations. This has to do with the fact that webpages hosting privacy policies often contain additional unrelated text and the fact that privacy policies are increasingly presented in a layered format. Additional challenges include URLs pointing to the wrong page or even pages that do not exist. We discuss the development and testing of an enhanced scraping tool that leverages generative AI to interact with web pages and scrape privacy policies more effectively. Starting from privacy policy URLs provided by service providers, we aim to collect the full text of privacy policies. We compare the performance of our technique with that of other tools proposed so far using a corpus of 275 privacy policy URLs for mobile apps in the iOS app store. The URLS were selected to include both popular and less popular apps. Our findings indicate that our proposed technique successfully retrieved a privacy policy from 90.2% of tested URLs from the iOS app store. Performance was compared against the Polipy tool and a naive scraper with Polipy's success rate measured at 77.8% and the naive scraper resulting in even lower performance. This research underscores the potential of generative AI to significantly improve the automation of privacy policy scraping.

1 Introduction

Although reading privacy policies has been one of the primary ways for users to understand how their data is collected and processed by the service providers, studies found that users rarely go through the entire content of these complicated policies since they are considered too long and difficult to read and comprehend [9][11]. To fill this gap, researchers have been developing natural language processing (NLP) techniques, such as policy summarization, privacy question-answering, and other tools to automatically analyze the content of privacy policies, aiming to help users understand the policies more efficiently and effectively over the past decades.

To analyze a privacy policy, the first step is to automatically extract the complete content of the policy from the webpage where it is published. However, sometimes it is challenging to get access to the webpage and accurately identify the privacy policy, as well as extract all the relevant content. Though developers are required to provide a URL to their privacy policy when releasing an app in the app store such as the iOS App Store [1] and Google Play [3], such URL may not always direct users to the page which includes the entire privacy policy. Alternatively, the URL may refer to the organization's homepage, other irrelevant webpages, or an error page with ”404 Not Found” or a DNS lookup error. These issues have been addressed by existing works [6] [14] [15] focusing on analyzing privacy policies for a large scale of apps from the app stores. To solve these issues, proposed solutions included training a classifier to decide if a webpage contains the privacy policy and discard all webpages that do not contain one to ensure all webpages they retrieved are privacy policy pages.

Besides irrelevant webpages, another issue that hasn't been widely addressed is incomplete privacy policy. Since most privacy policies are designed for human readers instead of automatic scrapers, human-friendly practices like linking less important information to other pages or showing a brief privacy notice instead of the full version are widely used. While users can get more information by interacting with the user interfaces (UI), these practices bring additional complexities for automatic tools to get the complete content. In addition, different service providers have different preferences in displaying ”less important” information. This may lead to a consistency issue when extracting a large amount of privacy policies. For example, some companies put information related to California Consumer Privacy Act (CCPA) at the end of the website displaying their privacy policies, while others choose to put them on another website with a link from the privacy policy. For human readers, these two practices are equally accessible, but automatic extractors may ignore information that is not explicitly displayed on the privacy policy webpage.

If an automated tool can decide which link in a given webpage is highly likely to contain a privacy policy or complement information related to privacy, it can access and retrieve more privacy policies compared to existing methods. In this report, we present our work that breaks down the challenges for automated tools to access complete privacy policies, and propose PoliScraper, a new privacy policy scraper using generative AI techniques to direct to the page with privacy policy from any given URL and to extract all information related to privacy policy on that webpage for further analysis. [1]

[1] Code is available at: https://github.com/TongJ05/PrivacyScraper

2 Background and Related Works

2.1 Existing Challenges when Automatically Extracting Privacy Policies

Previous studies have shown that a large fraction of URLs that service providers claim to contain privacy policies often fail to do so, leading users to incorrect or irrelevant pages instead. Alamri et al. analyzed [7] all apps from the iOS app store and found that more than half of the provided URLs failed to link to the privacy policy as expected. In a study on analyzing the complexity of mental health apps' privacy policies, Powell et al. [12] reported 22.5% of the URLs provided by the mental health apps in Google Play were not able to link to a privacy policy.

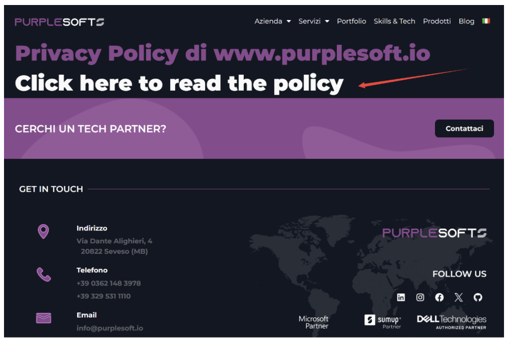
In practice, it is found that common destinations of these provided URLs are homepages of the company, broken or dead links ("404 Not Found"), or irrelevant webpages [14]. Therefore, in this report, we conducted a further analysis on the destinations of the problematic URLs and further proposed an approach to automatically navigate to the page containing the privacy policy from a given URL. More specifically, if a URL does not directly link to the privacy policy page, the automated extractor can distinguish the problematic situation from the four cases identified to provide a more concrete result correspondingly. Figure 1 shows an example of a webpage for each problematic situation.

Case 1: Indirect URL In this case, although the provided URL does not direct to a privacy policy, users can refer to another link on that webpage which will lead to the expected privacy policy page. Figure 1a shows an example: although humans may know that they could click on the hyperlink ("Click here to read the policy") to access the privacy policy, automatic privacy policy scraping tools may treat this navigating page as a non-policy page and discard this page, which could have been included in the dataset.

Case 2: Additional information displayed via link/UI In this case, the webpage contains a partial privacy policy or a simplified version of the privacy policy of the app. Users need to click on links or buttons to find more detailed information. Figure 1b depicts an example of this case: the webpage only displays part of the privacy policy, and users need to click on the "CCPA" link to access detailed information on the CCPA compliance.

Case 3: Block by security mechanisms In this case, the website designer intentionally implemented some mechanisms to prevent automatic tools from getting full text of the privacy policy easily. An example security mechanism is depicted in Figure 1c: the developers put their privacy policy in Microsoft OneDrive and ask users to sign in to access it. If users have already signed in somewhere else, they can access the privacy policy without seeing this sign-in page, but automatic tools, which do not have the sign-in functionality, will be blocked by the shown page.

Case 4: Irrelevant pages or page not found In this case, the URL provided by the developer leads to an expired webpage, an irrelevant webpage, or a page that does not exist. Figure 1d is an example of this case: the URL directs the user to a "404 Page not found" error page.


PURPLESOFTS

Azienda Servizi Portfolio Skills & Tech Prodotti Blog

Privacy Policy di www.purplesoft.io

Click here to read the policy

CERCHI UN TECH PARTNER?

Contattaci

GET IN TOUCH

Indirizzo
Via Dante Alighieri, 4
20822 Seveso (MB)

Telefono
+39 0362 148 3978
+39 329 531 1110

Email
info@purplesoft.io

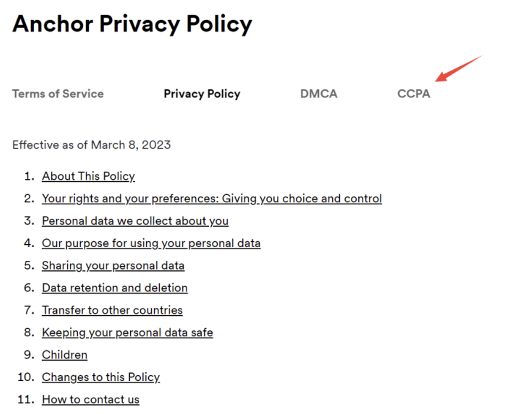
FOLLOW US

Microsoft Partner

sumup Partner

DELL Technologies AUTHORIZED PARTNER

(a) Case 1: Indirect URL

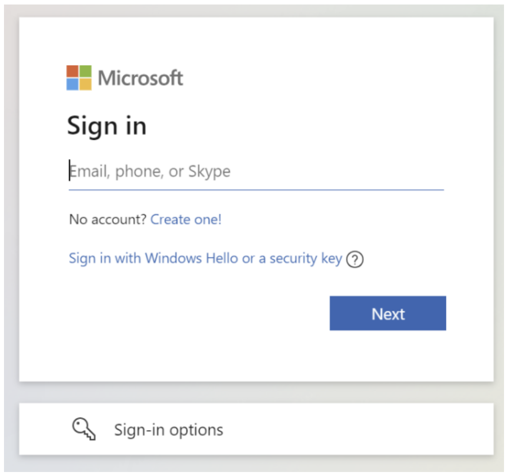

Anchor Privacy Policy

Terms of Service Privacy Policy DMCA CCPA

Effective as of March 8, 2023

1. About This Policy
2. Your rights and your preferences: Giving you choice and control
3. Personal data we collect about you
4. Our purpose for using your personal data
5. Sharing your personal data
6. Data retention and deletion
7. Transfer to other countries
8. Keeping your personal data safe
9. Children
10. Changes to this Policy
11. How to contact us

(b) Case 2: Additional information via link/UI


Microsoft

Sign in

Email, phone, or Skype

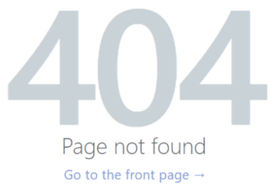
No account? Create one!

Sign in with Windows Hello or a security key

Next

Sign-in options

(c) Case 3: Security mechanisms


404

Page not found

Go to the front page →

(d) Case 4: Irrelevant/404 page

Figure 1: Cases that a URL failed to link to a privacy policy

2.2 Available Tools for Extracting Privacy Policies

We are aware of a number of libraries that can be used to download text from a given URL, including Boilerpipe [2], Html2text [5], and Trafilatura [4]. However, these libraries are not specifically designed for downloading text of privacy policies and thus do not optimize for the four mentioned cases. Considering the specific design practices, Polipy [13] is the library that is the best fit for extracting privacy policies. Besides using existing libraries, task-specific privacy policy extractors have also been implemented by researchers to retrieve relevant data [6] [14] [15].

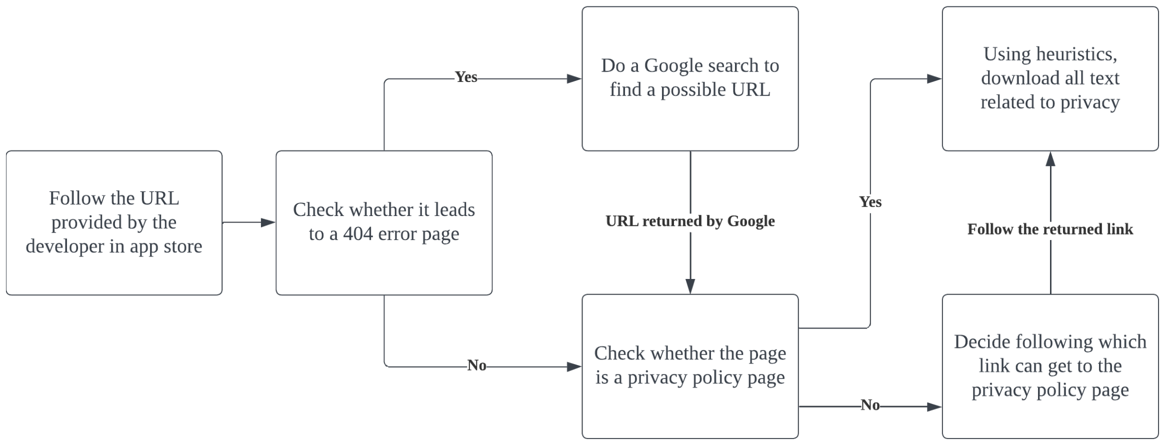

Follow the URL provided by the developer in app store
Check whether it leads to a 404 error page
Yes
No
Do a Google search to find a possible URL
URL returned by Google
Check whether the page is a privacy policy page
Yes
No
Using heuristics, download all text related to privacy
Follow the returned link
Decide following which link can get to the privacy policy page

Figure 2: A diagram of the collecting pipeline

3 New Solution: Utilizing Generative AI

Considering the capability of scrapers to navigate to other pages to access the full privacy policy, we designed PoliScraper, a new scraper to get the complete privacy policy from a given URL, which is able to decide when and where to navigate by itself automatically. Based on a pipeline that behaves differently for various kinds of webpages (Section 3.1, 3.2 and 3.3), instead of simply downloading the full text in a provided URL, PoliScraper can accurately identify the four kinds of problematic URLs discussed in the previous section. In addition, PoliScraper could access the full text of the privacy policy if the given URL includes a link to additional information related to privacy. To scrape and download the full content of the policy, Selenium with a Chrome driver instead of a single HTTP request is used since it can capture dynamically loaded contents and follow redirects as mentioned in [6] and [7]. This approach can successfully load contents in expandable UIs, thus partially solving the problems raised in Case 2. It also makes the scraper more human-like and has the potential to pass the security mechanisms built in the webpages mentioned in Case 3. See Figure 2 for details on the pipeline of our PoliScraper.

3.1 Step 1: Identifying Error Pages

Given a URL provided by a service provider that is expected to direct to the privacy policy, the first step is to check if the URL leads to an error page or a page that does not exist. This is because an error page typically contains limited information and does not include any link to a privacy policy. Such judgment can be made by training a machine learning model or prompting a generative language model. ChatGPT with chain-of-thought technique proposed by [10] is used in PoliScraper to make this decision. [2]. If the result shows that the URL leads to an error page,

[2]Two prompts are necessary when using chain-of-thought. They are: 1.”The following webpage is the content in a webpage: [WEBPAGE CONTENT], Given the content of this website, please determine if the provided content corresponds to an error page or not. List up to 3 supporting evidence and think step by step. Limit the explanation of each evidence to 1 sentence. You must stand for either yes or no.” 2. ”In a word (Yes/No), the answer is:”

PoliScraper will try to figure out if there is another URL that may lead to the actual privacy policy. Given the fact that the only available information is the name of the app in this case, Kumar et al. [8] proposed a solution to locate the privacy policy: conducting a Google search using a query ("[APP NAME] + privacy policy English"), and this method is adopted in this study. Meanwhile, if the page is not an error page (not problems mentioned in Case 4), PoliScraper will continue proceeding to the next step (Section 3.2) to analyze the provided URL.

3.2 Step 2: Navigating to the Privacy Policy Page

While the first step solves problems in Case 4 with an output of a URL that may lead to a privacy policy page and does not lead to an error page, the next step will try to resolve the problems mentioned in Case 1 (indirect URLs). First, generative AI will decide whether the URL from the previous step leads to a privacy policy page. [3] If the page is a privacy policy page, PoliScraper proceeds to the next step (Section 3.3). Otherwise, it will try to navigate to the privacy policy page by choosing a link on that page to follow. After collecting the anchor text associated with each link, generative AI will decide which link to follow. [4] By following the link that generative AI has chosen, PoliScraper now navigates to a privacy policy page. If no link is considered to be valid, all contents on the current page will be recorded, and the scraping process will be marked as completed.

3.3 Step 3: Collecting Full Text from the Privacy Policy Page

The previous step has provided a webpage containing at least a partial privacy policy. In this step, PoliScraper first downloads everything on that webpage, and then tries to retrieve all information related to privacy policy from that page to identify and resolve problems mentioned in Case 2. We noticed that most URL links for additional information are related to the California Consumer Privacy Act (CCPA) and General Data Protection Regulation (GDPR), thus the following heuristics are used to analyze the anchor text associated with a link to determine whether it contains additional information that should be collected. An anchor text is considered related if it contains (case insensitive):

[3]The following two prompts are used: 1."The following webpage is the content in a webpage: [WEBPAGE CONTENT], Please determine if the content of the webpage contains the beginning of a privacy policy (Terms of uses are not privacy policies). If the webpage offers links to the privacy policy, return "No" directly. This is the top priority. After considering this, if it is a random webpage or the homepage of the software, return "No". If it is the beginning of a privacy policy, return "Yes". List up to 3 supporting evidence and briefly explain. Limit the explanation of each evidence in 1 sentence. You must stand for either yes or no. 2. "In a word (Yes/No), the answer is:"

[4]In this step, chain-of-thought is not used, and the following prompt is used: "The following contents are anchor texts associated with links on a website: [ANCHOR TEXT LIST], According to the previously provided information, I want to navigate to a website containing a company's privacy policy, and now I am in a website that may have a link to my destination. Please decide clicking which link can take me to the company's privacy policy page. If there is no possible link, output NONE. If there is a possible link, output the anchor text only. If there are multiple possible links pointing to policies in different languages, output the anchor text to the English policy only. Do not use complete sentences when responding."

((("california" OR "ca") AND ("notice" OR "privacy")) OR("ccpa")) OR (("european union" OR "eu") AND ("privacy" OR "notice"))

After collecting the content of each webpage with additional information, the collected text is appended to the end of the original privacy policy collected at the beginning of this step.

4 Evaluation

To evaluate how well PoliScraper performs with URLs that service providers claim to contain a privacy policy in app stores, samples from Apple's App Store were chosen to be the test set. Apple's App Store was selected because of the rich information provided, given it published an alphabetical list of all available iOS apps with some of them being classified as popular apps. Drawing inspiration from [6], it was recognized that this classification is significant and enables the analysis of the performance on both random apps (which serve as a representative sample) and popular apps (which tend to be more prevalent on users' devices and gain more attention).

4.1 iOS App Sampling and Test Set Construction

On October 14, 2023, we crawled the iOS app store. We retrieved a list of 145,128 unique apps which provide a URL to their privacy policy, and 4,898 were classified as popular apps. Among 4,898 popular apps, 100 apps were selected randomly as sample popular apps. Among the remaining 140,230 apps, 200 apps were selected randomly as sample random apps. After identifying these 300 sample apps, each app was manually checked by the researcher to see if the provided URL directed to a privacy policy page. For URLs that did not link to a privacy policy page, we manually categorized them into one of four pre-defined cases. In instances where the URL did not direct to a privacy policy page, attempts were made to ascertain whether the app had a privacy policy somewhere else. It was found that 3 out of the 100 sampled popular apps and 22 out of the 200 sampled random apps did not have a privacy policy, despite having a URL listed in the app store. Consequently, the final test set included 275 apps, with 97 popular apps and 178 random apps.

4.2 Evaluation Metrics

The following performance of PoliScraper was evaluated using the test set: 1) the ability to distinguish whether a webpage is an error page or a privacy policy page, and 2) the ability to download the full privacy policy given a URL that may lead to a privacy policy in the iOS app store. In this study, we utilized GPT 3.5 as the generative language model and all results were based on GPT 3.5 on November 14, 2023.

A naive approach was also implemented to identify the simplest method as a baseline. Such naive approach used Selenium and a headless Chrome driver and downloaded all the content displayed in the provided URL directly. In addition to this approach, the performance of Polipy [13], a library designed for downloading and analyzing privacy policies, was also evaluated with the test set as an additional baseline.

4.3 Performance on Classifying Pages

Classifying if a webpage is an error page is a relatively easy task for PoliScraper with a satisfying result: all error pages were correctly identified and other pages were never misclassified as error pages in both 97 popular apps and 178 random apps. (100% accuracy and recall are achieved in step 1)

The classification on the privacy policy pages could be more challenging, and some misclassifications exist in step 2. For the 97 popular apps, the precision is 97.26% and the recall is 86.59%. For the 178 random apps, the precision is 95.56% and the recall is 93.48%. To further investigate GPT-3.5's ability to distinguish privacy policy pages and other legal statement pages, we collected URLs to the following three types of pages and let GPT-3.5 decide if the given page is a privacy policy page: ”terms of use”, ”legal notices”, and ”random page”. [5] The fraction of apps to be classified as privacy policies for these three kinds of confusing pages is 12.5%, 10%, and 0% respectively. We also found that a better-performing language model is not confused by these pages that look similar to a privacy policy page: for GPT-4, none of these three kinds of pages are classified as a privacy policy page (tested on November 14, 2023).

4.4 Performance on Downloading Complete Privacy Policies

We also tested the ability to collect full privacy policies in practice, with results in Table 1. Note that for this study, only downloaded text is taken into account in our evaluation. In other words, if a tool got all the expected text (the full privacy policy) for a given URL, we considered it as a successful case regardless of the format of the downloaded text.

4.5 Runtime Analysis

Table 2 provides the runtime of all three methods to extract a privacy policy from a single URL over the test set. According to the runtime of the Naive Approach, visiting the provided URL and extracting all its contents takes 1.82 seconds on average. For PoliScraper, it requires an additional 4.88 seconds for each URL on average to complete the ChatGPT API call, analyze the response, and navigate to another site to improve performance if necessary. For PoliScraper, a detailed runtime breakup of is provided. Specifically, the total time taken is segmented into three parts: 1) Selenium (create a Chrome driver, visit webpages, do a Google search if necessary), 2) ChatGPT (interact with ChatGPT API), and 3) Other (analyze ChatGPT's response, write extracted text to a file). From this breakup, waiting for the ChatGPT API to respond takes the most time.

[5]Detailed definitions of these three types of pages: 1. ”Terms of use”: The “terms of use” page of a random app. 2. ”Legal notices”: The “legal notices” page of a random app. Pages that merge “terms of use” and “legal notices” into a single page are not included. 3. ”Random page”: Pages containing contents other than “terms of use”, “legal notices”, and “privacy policies” in the website of a random app. Examples include the company's homepage and the production introduction page.

		Naive Approach		PolicyScraper		Polipy	
100 Popular Apps	Case 1: Indirect URL	0/4	0%	2/4	50%	0/4	0%
	Case 2: Additional info via link/UI	0/13	0%	12/13	92%	5/13	38%
	Case 3: Security mechanisms	0/8	0%	5/8	63%	0/8	0%
	Case 4: Irrelevant/404 page	0/3	0%	2/3	66%	0/3	0%
	Page with complete privacy policy	69/69	100%	67/69	97%	69/69	100%
	Total (3/100 don't have policy)	69/97	71%	88/97	91%	74/97	76%
		Naive Approach		PolicyScraper		Polipy	
200 Random Apps	Case 1: Indirect URL	0/3	0%	2/3	66%	0/3	0%
	Case 2: Additional info via link/UI	0/8	0%	8/8	100%	4/8	50%
	Case 3: Security mechanisms	0/5	0%	4/5	80%	0/5	0%
	Case 4: Irrelevant/404 page	0/25	0%	9/25	36%	0/25	0%
	Page with complete privacy policy	137/137	100%	137/137	100%	136/137	99%
	Total (22/200 don't have policy)	137/178	77%	160/178	90%	140/178	79%

Table 1: This table provides the distribution of cases and the performances for each technique.

4.6 Discussion

According to the results shown in the previous sections, our approach shows the ability to download more privacy policies than the Naive Approach and Polipy in a reasonable amount of time. The performance of our approach is improved by the ability to navigate to other pages. Among 100 popular apps, only 69 of them provide the complete privacy policy that an automatic tool can easily access with the URL provided in the iOS app store, and this fraction is similar for random apps (138 out of 200). Most existing privacy policy extraction methods discard URLs that do not directly lead to a privacy policy. However, for some of these discarded URLs (Case 1 and Case 2), clicking on some link on the page is the only action needed to navigate to a privacy policy page or access additional information related to privacy, and our approach would help with proceeding such actions automatically, thus identifying and retrieving more privacy policy from the URLs.

Another observation is that URLs in the listing page of popular apps and random apps are different in terms of how they are problematic. For popular apps, the major problem is that they tend to use human-friendly techniques that require interaction (Case 1 and Case 2) to display their privacy policies instead of displaying the full static privacy policy document. Security mechanisms against automatic tools (Case 3) are more common for popular apps as well. For random apps, displaying a static privacy policy document is the common practice and the major problem is that many apps give a broken URL to their privacy policies (Case 4) although some of them have a working privacy policy webpage: among these 47 cases, 25 apps have a privacy policy published somewhere else.

Utilizing generative AI models to inform decision-making and guide privacy policy extraction introduces a question: what are the implications of an incorrect decision? Our approach tolerates false negatives in deciding if a page is a privacy policy page. If a privacy policy page is classified

	Average (s)	Minimum (s)	Maximum (s)
Naive approach	1.82	1.16	2.83
Polipy	15.85	12.47	25.74
PoliScraper	6.70	5.05	43.97
PoliScraper - Selenium	2.17	1.19	2.91
PoliScraper - ChatGPT	4.52	3.81	42.03
PoliScraper - Other	0.01	0.01	0.02

Table 2: This table provides the runtime for each technique over the test set (in seconds). For PoliScraper, the last three rows provide a runtime breakup.

as a non-policy page, the next step is to find links on that page that may lead to a privacy policy page. If there is no possible link, we will record everything on the current page, which is a privacy policy. As a result, the privacy policy can still be downloaded though the classification is incorrect in practice.

A limitation of our approach is that it is not robust towards false positives in deciding if a page is a privacy policy page. If a non-policy page is classified as a policy page, everything on that page will be downloaded and this gives a non-policy page. Since generative AI tools cannot give a 100% precision (precision is 97.26% for popular apps and 95.56% for random apps in our experiment), some non-policy pages will be downloaded. Another limitation is that we used ChatGPT API as the generative language model for its high performance, but it introduces more randomness to the pipeline as well because its performance will change over time. It also results in a stability issue according to Section 4.5. Though PoliScraper is faster than Polipy on average, using ChatGPT API results in some extreme cases that the time taken to process a single URL is longer than 30 seconds. Replacing it with a more stable generative language model with a similar performance may be a future direction.

5 Conclusion

Service providers are typically required to provide a URL pointing to their privacy policies. Such is the case in popular app stores including the iOS app store and Google Play Store. Many regulations around the world mandate the presence of privacy policies. Yet, URLs do not always point to actual privacy policies and extracting the text of these policies has proven to be challenging given the presence of additional text on many pages as well as the adoption of layered presentations where accessing the entirety of the text of a policy requires following multiple links. In this report, we identified and addressed four different sources of difficulty when it comes to scraping the text of privacy policies. We introduced PoliScraper, a new scraper that aims to better handle these 4 sources of difficulty in contrast to other tools. In a test set containing 275 URLs for apps from the iOS app store, PoliScraper was shown to accurately extract the full text of privacy policies 90.2% of the time. By overcoming the obstacles in extracting privacy policies automatically and allowing the scraper to do necessary navigation, PoliScraper seems to be able to more effectively scrape privacy policies than previously proposed tools. Given the importance of being able to scrape the text of privacy policies in all work relying on NLP to analyze the text of policies, we hope that our new tool will contribute to further progress in this area.

References

[1] App store review guidelines. `https://developer.apple.com/app-store/review/guidelines/#legal`. Accessed: 2024-02-23.

[2] Boilerplate removal and fulltext extraction from html pages. `https://github.com/kohlschutter/boilerpipe?tab=readme-ov-file`. Accessed: 2024-02-23.

[3] Prepare your app for review. `https://support.google.com/googleplay/android-developer/answer/9859455?hl=en&visit_id=638443132878662986-3226506662&rd=1`. Accessed: 2024-02-23.

[4] A python package command-line tool to gather text on the web. `https://trafilatura.readthedocs.io/en/latest/`. Accessed: 2024-02-23.

[5] Turn html into equivalent markdown-structured text. `https://pypi.org/project/html2text/`. Accessed: 2024-02-23.

[6] Jose M. del Alamo Norman Sadeh Akshath Jain, David Rodriguez. Atlas: Automatically detecting discrepancies between privacy policies and privacy labels. *International Workshop on Privacy Engineering*, 2023.

[7] Hamad Alamri, Carsten Maple, Saad Mohamad, and Gregory Epiphaniou. Do the right thing: A privacy policy adherence analysis of over two million apps in apple ios app store. *Sensors*, 22(22), 2022.

[8] Vinayshekhar Bannihatti Kumar, Roger Iyengar, Namita Nisal, Yuanyuan Feng, Hana Habib, Peter Story, Sushain Cherivirala, Margaret Hagan, Lorrie Cranor, Shomir Wilson, Florian Schaub, and Norman Sadeh. Finding a choice in a haystack: Automatic extraction of opt-out statements from privacy policy text. In *Proceedings of The Web Conference 2020*, page 1943–1954, New York, NY, USA, 2020. Association for Computing Machinery.

[9] Fred H. Cate. The limits of notice and choice. *IEEE Security Privacy*, 8(2):59–62, 2010.

[10] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023.

[11] Aleecia M. McDonald and Lorrie Faith Cranor. The cost of reading privacy policies. 2009.

[12] Adam C Powell, Preeti Singh, and John Torous. The complexity of mental health app privacy policies: A potential barrier to privacy. *JMIR Mhealth Uhealth*, 6(7):e158, Jul 2018.

[13] Kothari S. Siyed Z. Wijesekera P. Fischer J. Hoofnagle C. Samarin, N. and S. Egelman. Investigating the compliance of android app developers with the ccpa.

[14] Sebastian Zimmeck, Peter Story, Daniel Smullen, Abhilasha Ravichander, Ziqi Wang, Joel R. Reidenberg, N. Cameron Russell, and Norman M. Sadeh. Maps: Scaling privacy compliance analysis to a million apps. *Proceedings on Privacy Enhancing Technologies*, 2019:66 – 86, 2019.

[15] 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. 2017.