Parsing Privacy Policies to Identify Cookies

CS 499 Final Report

Anay Singh Sisodiya
21110022, sisodiyaanay@iitgn.ac.in

Darshi Gaurang Doshi
21110050, doshidarshi@iitgn.ac.in

Supervised By: Prof. Abhishek Bichhawat, Gayatri Priyadarsini (co-mentor)

1. Abstract

We are automating the extraction and categorization of cookie information from cookie policy pages. By employing a heuristic-based approach combined with natural language processing (NLP) techniques, the system identifies cookie tables, extracts relevant data, and classifies cookies into predefined GDPR categories. The output contains a structured CSV file with the website URL, extracted cookies, and their categories. While there are challenges like dynamic content and cookie representations in the form of pdf and images. In the case of the tables this heuristic approach offers a scalable and resource-efficient solution, making it applicable for large-scale cookie policy analysis across diverse websites. Future improvements include refining the system to handle edge cases and further scaling the solution. *The aim is to bridge the gap between regulatory intent and the ground reality by ensuring that websites adhere to GDPR cookie declaration guidelines[1].*

2. Introduction

In today's digital era, the increased internet connectivity raises concerns about personal data privacy along with bringing many benefits. Websites and corporations often collect large amounts of user data, tracking individuals and building detailed profiles, frequently without their clear consent. This raises ethical concerns and exposes users to potential misuse of their personal information. To address these concerns, the General Data Protection Regulation (GDPR) aims to protect individuals' data privacy by setting rules on the collection, processing, and storage of personal information. A key component of GDPR is ensuring that websites transparently declare and categorize the cookies they use. However, many websites still find ways to bypass the rules, and hence, data privacy often remains a secondary concern.

Suppose you visit an OTT platform, and a cookie consent banner or popup asks you to accept cookies to enhance your experience. They will show the options for accepting cookies categorized under necessary, functional, and advertising cookies. However, when you go deep into this cookie information, you will know the description under the cookie is vague. For example, under advertising cookies, instead of mentioning which companies are collecting data, they use broad terms like "third-party advertisers", keeping the users in the dark. This lack of

transparency prevents users from understanding which personal data companies collect, how they use it, and who has access to it.

3. Related Works

Various approaches have been developed to automate the analysis of website cookie policies. Our primary inspirations for this project are Cookie Banner by Santos et al., Cookiescanner by UBA-PSI, and Cookie Block by Adel Bouhoula.

Cookie Banner by Santos et al. [2] focuses on detecting cookie banners by manually creating a dataset of banners using the Tranco list. It identifies five key features of cookie banners: purpose, framing, misleading language, vagueness, and technical jargon. Cookiescanner [3] analyzes websites by loading their web pages in Chrome and employs various cookie banner detectors (e.g., NaiveDetector, BertDetector). It combines keyword detection with machine learning models like BERT to identify cookie consent notices and compare them against GDPR requirements. As Cookiescanner primarily operates on banners, it does not rely on additional search mechanisms.

Cookie Block by Bouhoula et al. [4] addresses several limitations of these approaches by automating and generalizing the analysis of website cookies. It uses a crawler powered by a BERT model to interact with cookie notices. Unlike the previous methods, it focuses on cookie notices appearing as website popups. Cookie Block employs a Depth First Search (DFS) approach to crawl websites, categorizing cookies into specific types by analyzing their descriptions, names, and attributes in detail using machine learning models. Additionally, it enables users to automatically reject cookies that do not match their preferences. While Cookie Block is highly effective and robust for most websites, the DFS approach is computationally intensive, potentially making it resource-heavy for some machines.

4. Cookie Policy Information

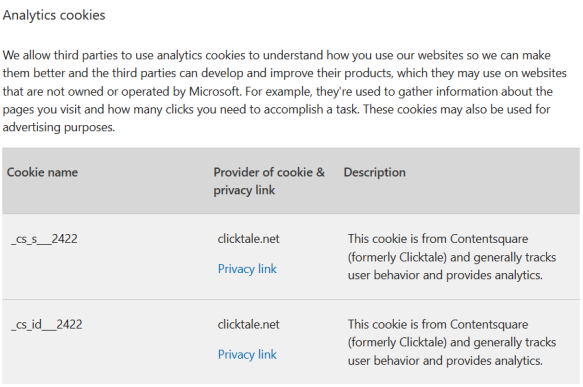
Websites use many approaches toward disclosing cookie-related information, which could make the extraction and categorization of data quite tricky sometimes. Not all websites have cookie information in structured tables, but the structure and nomenclature can be substantially different when they do. For example, some tables use an entire column to define categories, while others use separate sections for different information. Many websites also use JavaScript to load cookie data dynamically, which may be based on the user's actions like scrolling or even clicking. This dynamic loading is even more complex regarding automation, as access to this information may take some time.

The main issue with the two approaches mentioned is that none works for the cookie policy pages, which contain cookie information in tables. For tables, Muhammad Yusuf Hassan worked on an approach that utilized a BFS (Breadth-First Search) crawler with a maximum depth of 4

and used two modules, table handler and popup handler[1]. Still, their compatibility could be better as they work on fewer websites. We aim to develop an approach to handle popups and tables and evaluate the cookie consent using NLP techniques. It can also be compatible with a more significant number of websites.

Name	Category	Location	Description
STYXKEY-not_authenticated	Performance	.gandi.net	Cookie Varnish, used to manage the site's cache
sharing_id	Performance	.gandi.net	Organization selected
websession	Performance	.gandi.net	Session cookie NodeJS
sessions	Performance	.gandi.net	Session cookie Oauth2
oauth.[uuid]	Performance	.gandi.net	SSO (connection via id.gandi.net to all Gandi's "connected" subdomains)
temporary_cart	Performance	*.gandi.net	Anonymous cart identifier

(a) Classification of cookies within the table[5]



Analytics cookies

We allow third parties to use analytics cookies to understand how you use our websites so we can make them better and the third parties can develop and improve their products, which they may use on websites that are not owned or operated by Microsoft. For example, they're used to gather information about the pages you visit and how many clicks you need to accomplish a task. These cookies may also be used for advertising purposes.

Cookie name	Provider of cookie & privacy link	Description
_cs_s__2422	clicktale.net Privacy link	This cookie is from Contentsquare (formerly Clicktale) and generally tracks user behavior and provides analytics.
_cs_id__2422	clicktale.net Privacy link	This cookie is from Contentsquare (formerly Clicktale) and generally tracks user behavior and provides analytics.

(b) Cookie classification in prev text[6]

Figure 1: The Two Main Formats of Presenting Cookie Classifications in Cookie Policy Pages.

5. Initial Approaches

We started by looking at the code from Muhammad Yusuf Hassan's work, which is comparable to Cookiescanner's methodology. This code created a BFS (Breadth-First Search) crawler that handled popups and tables using particular modules. Although it gave us a basic grasp, we discovered it was only compatible with a limited number of websites.

The Cookie Block by Adel Bouhoula et al., which employs a Depth-First Search (DFS) crawler coupled with machine learning models like BERT, was then replicated and evaluated. But the setup procedure was complicated. The tool was resource-intensive, resulting in performance problems, and each step required installing new software or libraries. Despite finishing the setup, the system could not parse several well-known websites.

In order to overcome these problems, we began creating our crawler. Our crawler was able to locate and access popups, but we had a lot of trouble navigating through them. Major issues we encountered as below

- Interactivity Issue: Our crawler was able to identify buttons within popups, these buttons lacked interaction. Consequently, the crawler could not initiate the necessary operations to retrieve the concealed cookie data.
- Hidden Data: Most cookie information was in hidden parts, and we manually needed to click them to see details. These interactions were complicated for our bot to simulate accurately.

The inability to automatically trigger and interact with cookie popups was a common problem across all implementations. Although we were able to collect links to cookie-related sections, a significant obstacle still existed when it came to triggering the necessary elements within popups.

6. Methodology

Unlike previous works that primarily focus on cookie banners or popups, our approach centers on extracting and analyzing cookie tables within cookie policy pages. Cookie tables often contain structured information about cookies, their categories, and their purposes. We began by manually analyzing several cookie policy pages to understand the common structures and patterns within these tables. During this analysis, we observed that some tables explicitly contained a column for categories, while others only provided cookie names and their purposes, requiring additional context to infer their categories.

6.1 Heuristic and Similarity Based Logic

Based on these observations, we formulated heuristics to identify cookie tables and extract the required information. The extraction process relies on keyword matching, using terms like "cookies", "type", and "category" to locate relevant columns in the table. Additionally, we incorporated a contextual analysis mechanism for cases where the "Category" column is missing or unclear. This approach leverages the previous text, *prev_text*, the text preceding the table, which often describes the purpose of the cookies listed, providing crucial context for classification. By employing NLP techniques such as similarity analysis, the system matches the prev_text to predefined GDPR-compliant cookie categories (e.g., Necessary, Functional, Advertising, Analytics). This ensures accurate classification, even in the absence of explicitly stated categories in the cookie table.

Category	RegEx for various Keywords
Necessary	essential, mandatory, necessary, required
Functional	function*, preferenc*, secure, security, authenticat*, video
Analytics	anonym*, analytic*, measurement, performance, research, statistic*
Advertising	ad, advertis*, ads, ad selection, fingerprint*, geolocation, market*, personalis*, personal info, sale of data, target*, track*, social, media
Unclassified	uncategorize*, unclassified, unknown

Table 1: Regular Expressions to match different categories

6.2 Implementation Details

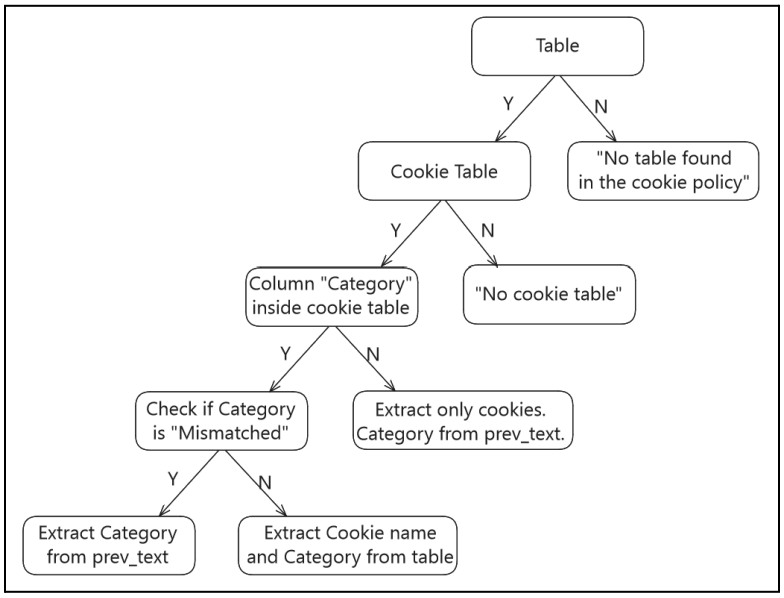


Figure 2: Roadmap to categorizing cookies

The decision-making process for categorizing cookies is outlined in the flowchart in figure 1. The process begins by checking if a table exists on the webpage. If no table is found, the system logs that no table is available. When a table is identified, it is further evaluated to determine if it qualifies as a cookie table, based on predefined heuristics. If no cookie table is identified, the system logs "No cookie table found."

```
# Utility Functions
Define is_similar(header, keywords) -> Check header similarity using TF-IDF
Define getPrevText(table) -> Extract preceding text for classification context
Define classifyCookieText(text) -> Match text with regex patterns for classification

# Main Table Handler
Define runTableHandler(url, html):
     Parse HTML for <table> elements
     For each table:
          Convert to DataFrame
          Identify cookie_col(cookie names) and category_col(categories) using
          is_similar()
          If both columns exist:
              Classify rows by category or preceding text
          Else if only cookie_col exists:
              Extract cookie names and classify using preceding text
     Return classifications
```

For valid cookie tables, the next step is to check for the presence of a "Category" column. If this column exists, the system proceeds to analyze the categories. In cases where the categories are ambiguous or mismatched (e.g., vague terms like "Miscellaneous"), the system falls back on the prev_text to provide more accurate classification. If the categories are clear, the cookie names

and categories are directly extracted from the table. On the other hand, if no "Category" column exists, only the cookie names are extracted from the table, and the prev_text is used to infer their categories.

7. Result and Conclusion

The final output of this process is a CSV file with five columns: url (the URL of the analyzed webpage), cookies (names of the extracted cookies), prevText (preceding text used for classification), prediction (the determined category of each cookie), and match (indicating whether the extracted category aligns with expected GDPR categories).

About 50 websites with cookie policy pages were included in the dataset we started with. Certain websites use tables, while others use other structures to display cookie information. The dataset included various scenarios, including websites with dynamic content and those without tables. Tables were not found on 15 of the 50 websites. There were several causes behind this:

- Some websites had login pages where the cookie policy information table was presented in popups. These pop ups dynamically load content, which was not accessible through our current approach.
- A few websites contained cookie information in PDF formats, which our table handler could not process correctly.
- Some websites had images representing cookie information rather than text-based tables, further complicating the extraction process.

We realized that heavy models, such as those based on deep learning, were impractical for our purpose because of these difficulties. These models are ineffective for the scale of this project since they demand a significant amount of time and computational resources. Instead, we used a heuristic strategy in conjunction with similarity checks based on natural language processing. Despite using fewer resources, this method worked well for managing various websites while preserving respectable scalability and performance.

8. Future Work

In the future, this algorithm can be modified to scale it to a lot more websites like the crux website dataset by addressing edge cases like cookie tables in unconventional formats, such as PDFs or images, by integrating an automated crawler using NLP algorithms to extract cookie policy pages from various websites. Detailed statistics can also be generated on the websites processed, such as how many were successfully parsed, which failed, and why. We can also analyze the distribution of cookies by category (e.g., Necessary, Analytics, etc) to provide insights into cookie practices across websites. Lastly, based on our collected dataset, a system can also be developed that can predict whether a website is correctly categorizing cookies and using them for their intended GDPR-compliant purposes.

9. References

[1] SEC-IITGN, “cookie_banner-table_compliance,” GitHub, [Online]. Available: https://github.com/SEC-IITGN/cookie_banner-table_complaince.
[2] C. Santos et. al., “Cookie Banners, What’s the Purpose? Analyzing Cookie Banner Text Through a Legal Lens,” arXiv, vol. 2110.02597, 2021. [Online]. Available: https://arxiv.org/pdf/2110.02597.
[3] UBA-PSI, “CookieScanner,” GitHub, [Online]. Available: https://github.com/UBA-PSI/cookiescanner/.
[4] Bouhoula, Ahmed. alsacnc. GitHub, [Online]. Available: https://github.com/bouhoula/alsacnc.
[5] https://www.gandi.net/en-US/contracts/privacy-policy
[6]https://support.microsoft.com/en-us/topic/third-party-cookie-inventory-81ca0c3d-c122-415c-874c-55610e017a6a