ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Analyzing Cookies Compliance with the GDPR

Master Thesis

Dino Bollinger

March 16, 2021

Advisors: Karel Kubicek, Dr. Carlos Cotrini, Prof. Dr. David Basin
Institute of Information Security, ETH Zürich

Abstract

With the introduction of the *General Data Protection Regulation* (*GDPR*), cookie consent notices have become a widespread phenomenon across the web. They come in the form of popups and banners that ask the visitor whether they consent to the collection of personal data, tracking, and the setting of cookies in the visitor's browser, separated by individual usage purposes. While consent notices give the user the impression that they have control over their privacy and personal data rights, many websites use dark patterns to nudge and deceive users, and others outright ignore the user's privacy preferences.

We try to address this problem by developing a browser extension that can locally enforce a user's cookie consent choices regardless of how the website behaves. First, we perform a study to analyze how a select number of Consent Management Platforms (*CMPs*) store cookie consent labels. Then, we use this information to design a web crawler that can determine if a website uses a specific CMP, and can retrieve a dataset of consent category labels, each of which assigns a cookie to a personal data collection purpose.

We define an extensive set of feature engineering steps to extract information from cookies, taking into account attributes such as the expiration date and the structure of the cookie's content. These features, as well as the collected labels, are used to train a series of tree-ensemble classifiers, using efficient algorithms such as XGBoost. The resulting model automatically assigns browser cookies to four distinct usage purposes. Our best-performing approach achieves an overall accuracy of 87.21%, and outperforms the manually-classified cookie repository "*Cookiepedia*" in both precision and recall.

We then integrate the resulting predictor as part of a browser extension that we title CookieBlock. Using this extension, the user can specify their cookie consent choices once, upon which CookieBlock will automatically remove all cookies that the user did not consent to. This removes the need to trust third-parties, providing a fully client-side approach to enforcing cookie consent.

Finally, using the information gathered from the CMPs during the web crawl, we define six novel analyses that provide evidence for potential GDPR violations in the wild. These analyses are intended to assist authorities, researchers, and web-admins alike in determining whether a website is compliant with the GDPR.

Contents

Chapter 1

Introduction

Ensuring that the privacy of users is protected continues to be a challenging issue on the World Wide Web. While many people are concerned about the security of their personal data, most do not invest the time to read privacy policies thoroughly or determine which parties use which data for what purpose [29, 33]. Tracking and personal data collection are ubiquitous [19], and the collected information is considered a valuable asset in the advertising industry. In some cases, it is even sold for profit to other interested third parties [31]. The collection of such data without the user's explicit consent is a big privacy concern, as it may contain sensitive information, including the identity, geographical location, and political orientation of the user. As history has shown, this can be used for more nefarious purposes than just the personalization of advertisements, such as the manipulation of entire demographics in a targeted manner [5].

Historically, privacy laws such as the *ePrivacy Directive* [21] and the *Directive 95/46/EC* [20] have attempted to improve this state of affairs by defining guidelines and requirements that businesses and website hosts should follow when collecting user's personal data. However, such legislation has often fallen short of its goal, as it was frequently hampered by inconsistent implementation across EU member states [46]. This prevented them from taking widespread effect on the behavior of third parties [52].

More recently however, the *General Data Protection Regulation* (*GDPR*) of the *European Union* (EU) [22] has caused waves across the internet. Having gone into effect in May 2018, it repealed and replaced the Directive 95/46/EC and became immediately enforceable in all member states across the EU. By defining strict and enforceable legal requirements on personal data collection and tracking, it forced service providers and third parties to adapt to its rules within a 2 year grace period [2].

The GDPR applies not just to businesses and websites operating from within the EU, but also to all providers offering their services to EU citizens. In addition, a refusal to comply with the rules set up by the GDPR can lead to significant fines of up to 20 million Euros or 4% of a company's annual turnover [12]. Therefore, the effects of the regulation were observed globally [46]. Among its requirements it is stated that personal data collection requires a valid legal basis, such as *consent* (Article 6.1a). In Article 7 and Recital 32 of the GDPR, it is described that such consent must be *freely given, specific, informed and unambiguous* in order to be considered valid. Article 5(3) of the ePrivacy Directive states that this applies to all browser tracking technologies, including those involving browser cookies. And indeed, as cookies are one of the most ubiquitous tools used on the web, the most observable response to the GDPR can be seen in the increased popularity of so-called *cookie consent notices* and the services that offer them [16].

A consent notice serves to inform the user about the website's privacy policy, the tracking technologies used, and what data is collected. It displays the involved third-parties, and what purposes the cookies serve. Most importantly, a consent notice allows the visitor to consent to individual usage purposes. The presence of a consent notice, if set up in a satisfactory manner, can allow a website to comply with the GDPR and similar legislation.

Many website hosts however lack the detailed know-how necessary to implement a fully compliant approach, both from a legal and technical perspective. As a result, many websites instead take advantage of fully-fledged solutions offered by so-called *Consent Management Platforms* (*CMPs*), which offer consent notice plugins that handle the entire process automatically, and offer a multitude of configuration options [25, 56].

Prior work by Degeling et al. [16] has shown that the GDPR has lead to an increase of 16% in the number of cookie banners shown to users immediately after it went into effect. Libert et al. [35] have found a drop of 22% in the number of cookies set without user consent, and Urban et al. [54] find a 40% reduction of third-party tracking and sharing connections after the GDPR. It is hence apparent that the GDPR had a noticeable effect on the state of web privacy, allowing the user much greater control than before.

On the other hand, prior research has also observed that the issue of user privacy is still far from solved. The worldwide adoption of consent notices is still very low, with even leading CMPs only reaching a market share of around 1% worldwide [8, 16, 25]. Reading privacy policy statements can be incredibly time-consuming [39], and many users have become fatigued by the high frequency at which consent notices appear across the more popular websites [9]. This fatigue causes users to seek out the nearest button available to remove the consent notice, without considering what privacy impact clicking said button might have [1, 4, 24].

Dark patterns [6] are a widespread practice that seeks to exploit this behavior. A study by Utz et al. [55] has shown that 57.4% of 1000 examined websites utilize nudging, a form of influencing the visitor to give affirmative consent by highlighting the "*Accept All*" button, or hiding the option to reject consent for individual usage purposes. Machuletz et al. [37] find that most users which have been nudged towards accepting all cookies often cannot recall their decision, and express regret upon being informed of their choice. This raises the question of whether the consent that is being collected by consent notices can really be considered "informed".

Furthermore, while CMPs purport to fulfill the legal requirements set forth by the GDPR, in practice many implementations still violate even the most basic rules. Nouwens et al. [44] have shown that 88.2% out of 680 examined websites fail at least one of three simple conditions, including the requirements of *opt-in choices* and *explicit consent*. Matte et al. [38] have found that out of a sample of 1426 selected websites, 9.89% register affirmative consent before the user even makes a choice, 2.66% do not provide the means to reject any cookies, and 1.89% register positive consent even if the visitor explicitly rejected the cookie purpose.

These practices make the supposed control over a user's personal data appear rather illusory. While high-profile actors do face fines for their misbehavior [30, 43], authorities currently lack efficient means to verify whether a website complies with existing laws. Many of the rules, specifically in regards to cookie consent, require at least some degree of manual inspection, or may even be impossible to verify without extensive user studies [47]. This renders the detection of potential violations at scale difficult.

Our Contributions

The concerns formulated in the previous section motivate the idea of a consent enforcement mechanism that is located entirely at the client. Rather than needing to trust the individual service providers to respect the user's privacy, it would be preferable for the client to be able to specify his privacy preferences once, after which they are enforced on any website they subsequently visit.

Within the scope of this master thesis we developed CookieBlock, a browser extension that uses a novel approach to automatically assign cookies to consent categories. It rejects cookies that do not align with the user's chosen consent preferences, thus preventing personal data collection or user tracking if this purpose is not agreed to.

The basic idea of the approach is to apply techniques of supervised machine-learning to the domain of browser cookies. Rather than manually labelling each cookie, we instead analyze the implementations of various CMPs in

order to design a webcrawler that can automatically scrape websites for cookie purpose labels. Our webcrawler supports gathering data from the *CookieBot*, *OneTrust* and *Termly* CMPs, and we collected a dataset consisting of over 300 000 labeled cookie samples originating from over 26 000 websites.

Using this dataset, we then analyzed the properties and common contents of cookies, devising a selection of 54 different feature engineering steps to transform a cookie into a vector of numerical data. These features vectors are intended to adequately represent the complexity and patterns found in cookie data, and are used as the input to a series of existing classifiers.

We apply three different tree ensemble classifiers to the dataset, namely the XGBoost [10], LightGBM [32] and CatBoost [17] algorithms. Our best predictor achieves an overall average accuracy of 87.2%, and high precision and recall for the majority of categories. Notably, we achieve an average precision of roughly 94.97%, and an average recall of 90.25% on cookies used for advertising purposes.

To the best of our knowledge, we are the first to apply machine learning techniques to the domain of cookie data. As such, the only baseline we can compare our results to is the public cookie repository *Cookiepedia* [45]. It has been established in 2010 by *CookiePro*, and contains over 30 million recorded cookies with corresponding purpose labels, which were manually assigned by human operators. We query the repository for labels, compare these to the ground truth in our dataset, and thus construct a baseline performance that our classifier is set to beat. We find that the applied classifiers can outperform Cookiepedia in several categories of cookies, both in precision and recall, and achieve comparable performance in others.

We integrate the resulting predictor as the core component of a browser extension, which we title CookieBlock. A one-time setup asks the user to specify his preferred privacy policy. Then, whenever the browser receives a cookie, the extension automatically classifies it, and if the resulting category label is rejected by the policy, the cookie is removed from the browser. The user is also given the flexibility to set exceptions for individual domains that they trust personally. This provides fine-grained consent choices and does not require any third-party support to be implemented.

Because it is difficult for researchers and authorities to automatically assess whether a website host or CMP is fully compliant with the GDPR, we additionally demonstrate six novel analyses that produce evidence of potential GDPR violations. We find that out of 26 403 analysed domains:

- 15.46% list wrong purpose labels for *Google Analytics* cookies.
- 27.75% contain potentially wrong outlier labels, based on the majority opinion of other domains that declared the same cookie,

- 5.30% contain cookies with expiration times that are at least 50% longer than what was originally declared in the consent notice,
- 19.21% contain cookie declarations that are uncategorized,
- 86.08% contain cookies that were never declared,
- and 3.51% contain cookies with multiple, contradictory purposes.

Particularly surprising is the number of sites with undeclared cookies. A staggering 86.08% of all analyzed websites used at least one cookie that was never mentioned in the consent notice. This suggests that most hosts are not fully aware of all cookies that are present on their websites, or that these hosts fail to properly declare the correct name and domain for their cookies.

In total, 91.87% of all websites in our dataset show at least one of the above issues – or 49.30% if we exclude undeclared cookies from the analysis.

In summary, the major contributions of this thesis are:

1. Two webcrawlers that can efficiently collect cookie labels from websites using selected Consent Management Platforms, and a feature extraction process that extracts numerical data from collected cookies.
2. The training and evaluation of a series of tree ensemble classifiers to predict consent category labels for cookies found in the wild.
3. A browser extension that makes use of the resulting predictor, which can reject cookies based on their assigned labels and the user policy.
4. Six novel, semi-automated approaches that collect evidence for potential GDPR violations from a select number of CMP implementations.

Report Structure

The rest of the report is organized as follows: In Chapter 2, we provide some background information on cookies, the requirements of the GDPR and the emergence of CMPs. In Chapter 3, we describe the design of the web crawlers. In Chapter 4, we detail the feature engineering steps and how we implemented the classifier. In Chapter 5 we present the design of CookieBlock, our browser extension for enforcing cookie consent. In Chapter 6, we present the results of the crawl, the importance of the extracted features and the classifier performance. In Chapter 7, we present our methods to collect evidence for potential GDPR violations, and apply them to our own dataset. In Chapter 8, we discuss related work in the area of privacy policy enforcement and violation detection. Finally, in Chapter 9, we conclude the report.

Chapter 2

Background

2.1 Browser Cookies

Browser cookies are key-value storage pairs that can store text data of up to 4 kilobytes in size. They are uniquely identified by name, domain and path, and can normally only be accessed by the same host that is responsible for creating the cookie (same-origin policy). The main purpose of browser cookies is to keep track of website state, which would otherwise be lost when leaving the domain. Some cookie last only for the current session, while others are kept across sessions until an expiration date passes [41]. Some legitimate uses for cookies include:

- Remembering logins between browser sessions.
- Restoring shopping cart contents while browsing through a store.
- Keeping track of user-specific website style changes, e.g., dark mode.

Cookies can be constructed in one of two ways. Either they are sent to the client as part of an HTTP GET response when accessing a resource on the web, or they are created by JavaScript code running in the client browser. Cookies are always sent with every request to the domain that set them. As such, they can be used to recognize users on repeat visits if the cookie contained a unique identifier, or to collect personal data from the user.

Note that the amount of data stored in cookies is generally small to keep the bandwidth impact on the network request low, which makes the page load faster. As a result, cookies are not well-suited for general purpose data storage. A better fit for this purpose are the HTML5 local or session storage APIs. While these may also be used for storing personal data and for tracking purposes, in this work we will only consider browser cookies.

2.1.1 Cookie Purposes

Cookies are often used to implement service-critical functionality, including logins and shopping carts, or features that improve the user experience. However, they may also be used to track visitor actions. We distinguish between two types of tracking:

The first form we summarize under the umbrella term *"Web Analytics"*. Cookies of this type are used to track users only for the domain they are currently browsing. Examples of what is tracked includes user dwell time, the browser used to access the site, how many visitors arrive and from which regions. This is done to improve the website, but also to collect website performance measurements for marketing purposes. Tracking for this purpose can be anonymous, and does not require sensitive information. However, as we will see in the next section, the GDPR nevertheless considers these cookies to contain personal information, and they hence require consent.

The second form of tracking is performed across multiple websites, usually for the purpose of advertising and the sale of user data. Such cookies collect information on the browsing behavior and personal interests of the user in order to build a profile. This profile is most commonly used to have advertisements better match the interests of the tracked user. This can entail sensitive information, including habits, interests and political orientation of the person behind the computer, and therefore constitute the main privacy concern with cookies. Cookies that are used for this purpose are commonly hosted by third-parties that track users across multiple sites. However, not all third-party cookies necessary serve tracking purposes.

2.2 Data Privacy Law

The *General Data Privacy Regulation* (GDPR) is a data privacy law signed by the EU in 2016, which officially came into effect on May 25, 2018. It is an expansive document that covers many cases of personal data collection, even outside the World Wide Web. To keep this section succinct, we will only cover legal requirements that specifically apply to cookies.

In order for a website to collect personal data and track users, the host is required to establish a legal basis for doing so. Among the recognized valid legal bases specified in Article 6(a-f) of the GDPR are (a) *consent* of the visitor, (b) *contractual obligation*, (c) *legal obligation*, (d) *vital interests*, (e) *public interests* and (f) *legitimate interests* of the controller or third-party. Of these, consent and legitimate interest are currently the most common justification used for data collection [25]. As the name implies, consent notices are specifically focused on gathering explicit statements of consent from website visitors.

Article 7 and Recital 32 of the GDPR define the conditions for valid consent. Namely, consent must be *freely given, specific, informed* and *unambiguous*:

> *"Consent should be given by a clear and affirmative act establishing a freely given, specific, informed and unambiguous indication of the data subject's agreement to the processing of personal data relating to him or her, such as by a written statement, including by electronic means, or an oral statement."*

In addition, it clarifies:

> *"Pre-ticked boxes or inactivity should not therefore constitute consent."*

The consent request must also be posed prior to cookies being set. Its description must be intelligible, use clear language, and the consent controls must be easily accessible. Furthermore, the data controller must be able to prove that consent was given.

The gathered consent is invalid if any of the specified requirements are violated. If this occurs, the offending party may be liable for fines. Article 83 specifies that fines may be up to 20 million Euros or 4% of a company's annual turnover. Moreover, as per Article 3(1), the GDPR has extra-territorial scope. Not only does it affect websites and businesses that are located within the European Union, it also extends to all websites that offer their services to, and collect personal data from EU residents. This means that even websites that are hosted for example in the United States could be liable when violating the requirements set up by the GDPR.

Interestingly, the GDPR applies to all cookies that store some form of unique identifier, and not just those that collect sensitive data. This is described as follows in Recital 30 of the GDPR:

> *"[...] natural persons may be associated with online identifiers provided by their devices, applications, tools and protocols, such as internet protocol addresses, cookie identifiers or other identifiers [...]. This may leave traces which, in particular when combined with unique identifiers and other information received by the servers, may be used to create profiles of the natural persons and identify them"*

Some jurisdictions require consent for all cookies [11], with the exception of those that are strictly necessary for the provided service to operate. This exception is given by the *ePrivacy Directive* [21], which came into effect in 2009. It refers to Directive 95/46/EC [20], which has been repealed and replaced by the GDPR. Article 5(3) of the ePD states the following:

> *"Member States shall ensure that the storing of information, or the gaining of access to information already stored, in the terminal equipment of a subscriber or user is only allowed on condition that the subscriber or user concerned has given his or her consent, having been*

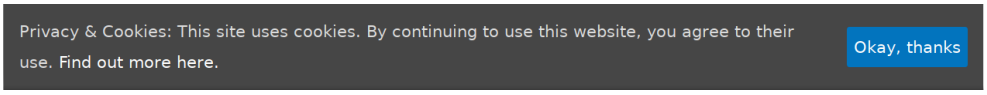

Figure 2.1: Example of a simple cookie banner. With the introduction of the GDPR, this type of consent notice has become insufficient and does not comply with the law's requirements.

> *provided with clear and comprehensive information, in accordance with Directive 95/46/EC, inter alia, about the purposes of the processing. This shall not prevent any technical storage or access for the sole purpose of carrying out the transmission of a communication over an electronic communications network, or as strictly necessary in order for the provider of an information society service explicitly requested by the subscriber or user to provide the service."*

The exception hence are cookies that are strictly necessary to enable services which the user asked for. These are specifically services that the visitor expects the service to provide. As such, a website host cannot simply declare any arbitrary cookie as being strictly necessary, and doing so could leave the host liable [30].

2.3 Consent Management Platforms

Consent notices are not a new phenomenon on the web. Design recommendations for them go back as far as 2001 [40], became widespread after the ePrivacy Directive came into effect in 2009. Consent notices come in many forms, including banners, prompts, overlays or pop-ups. The simplest examples, such as the one presented in Figure 2.1, only inform the visitor about the use of cookies, assuming the continued use of the site to be equivalent to consent, and offering no further options than to acknowledge the banner with a OK button [33].

However, when the GDPR was signed, it soon became apparent that existing consent notice implementations were no longer adequate. The requirement for "*specific consent*" demands cookies to be separated by individual usage categories. "*Informed consent*" means that each category had to be associated with a detailed description, and "*explicit consent*" required that agreement to data collection could no longer be implicitly assumed. In addition, consent needs to be communicated with potentially many separate third-parties, further complicating the process. As a result, the effort and cost to implement compliant solutions became substantially higher [25].

The result of this situation is what was aptly called "the commodification of consent" by Woods et al. [56], where so-called Consent Management Platforms (henceforth denoted as CMPs) offer fully-fledged consent management solutions to website hosts and businesses. By handling the legal terms

and conditions for privacy policies, and implementing complex consent dialogues, they allow the website to comply with regulations in exchange for a service fee. And indeed, CMP adoption has flourished in the years after the GDPR. In a work by Hils et al., the authors found that CMP adoption quadrupled between the years 2018 and 2020 [25].

A major player in the area of consent management is the *Interactive Advertising Bureau Europe*, which defined a advertising industry standard called the *Transparency and Consent Framework* (*TCF*). It is a specification through which websites can exchange the consent gathered from users with third-party vendors [38]. The CMP hereby acts as the intermediary, and implements the infrastructure necessary to perform this exchange. Many of today's leading CMPs are registered as part of this framework [26].

2.3.1 Dark Patterns

Despite the claims of offering fully GDPR compliant solutions, in many cases misbehavior can still be observed on websites that make use of CMPs. Utz et al. [55] analyse a random sample of 1000 CMPs, and find that 95.8% provide either no consent choice, or acceptance only, violating the requirement for specific consent. Nouwens et al. [44] show that 32.5% out of 2035 examined sites used implicit consent, and 56.2% of all sites used pre-selected options.

Nudging, while not explicitly forbidden, is a dark pattern [6] that is common practice in consent notice designs offered by CMPs. Utz et al. furthermore found that 57.2% of the examined websites attempted to guide users towards accepting the least privacy-friendly option, a strategy intended to benefit the hosts and advertisers, but not the visitor [55].

Finally, Matte et al. [38] find that some websites do not even respect the user's consent choices. They found that out of 560 examined websites which make use of CMPs that are part of the TCF, 54% of them contained at least one violation in where the user's choices were not reflected in the consent sent to third parties. This ranges from cases where consent was sent before the user made any choice, to instances where the user's choice was replaced by different preferences entirely.

Chapter 3

Cookie Consent Crawler

While retrieving a large number of cookies from the Web may be a simple task to perform, gathering labels for the purpose of classifying cookies is a topic that has so far largely been unexplored in scientific literature. One option would be to crawl arbitrary websites for cookies, and to then query the *Cookiepedia* repository [45] for associated purpose labels. Cookiepedia is a publicly accessible database of cookies, where each cookie is assigned to a specific purpose label and detailed description. The website has been established in 2010 by *CookiePro*, which is now part of *OneTrust*. They claim to store information for over 30 million cookies, which have been manually analyzed and classified by a group of human operators.

While using Cookiepedia as a basis would be a valid approach for gathering a set of labels, we instead tackle this task differently, by making use of the labels assigned to cookies by various Consent Management Platforms (CMPs). The advantage of this approach is that we can gather cookie labels directly from the hosts that created them, which provides data for unique cookies that the operators of Cookiepedia do not store in their database. Moreover, many CMPs keep track of a cookie's category and purpose in their own repositories, which are usually not publicly accessible. These repositories are used to suggest purposes and descriptions for third-party cookies that even the website owner which uses the CMP may not be aware of [13, 50]. In this fashion, we can gather a diverse set of data which furthermore allows us to run a number of analyses for potential GDPR violations, which will be examined in Chapter 7.

We acknowledge that the labels retrieved in this fashion may not always be entirely accurate. Because they are selected by a multitude of different human operators with varying levels of competency, human error or even intentional misbehavior may introduce noise into the collected dataset. As such, the collected dataset will have potentially unreliable labels, making this approach to classifying cookies an example of distant-supervision.

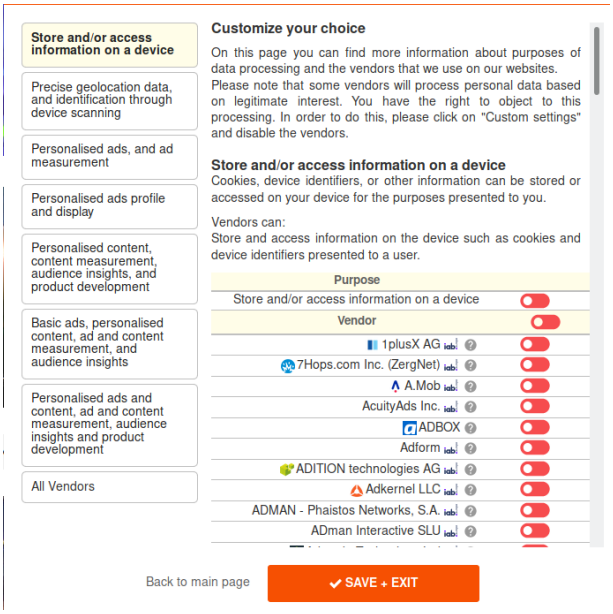

Figure 3.1: Example of a consent notice that offers an incredibly complex set of choices.

Note that it is difficult to determine a correct ground truth in the domain of cookies, as due to limited information, the actual purpose can often not be determined from the perspective of an outside observer. However, by comparing label assignments for equivalent third-party cookies, we are able to determine a lower bound of the degree of noise in the dataset. This will be explored in Section 6.1.4.

In the following, we will present our approach to gathering cookie category labels from CMP consent notices in the wild, detailing our selection process for CMPs and the web crawlers we designed.

3.1 Choice of Crawling Targets

Not every Consent Management Platform is a viable target for the purpose of retrieving cookie labels. The GDPR, for instance, does not specify strict rules on the structure of a cookie banner. Thus, website hosts are free to construct their consent notices in any fashion that is compliant with the requirements [2].

The direct result of this situation is that the space of consent notice design is large and greatly heterogeneous [37]. Even for websites that use the same CMP, the design of the consent notice can vary greatly, as customizability is often used as a selling point for such services. Those CMPs which offer only a rigid and fixed design, for example Cookiebot, even appear to be losing customers to their competitors [25].

Designs may range from simple banners appearing at the bottom of the page, to multi-layered popups covering the entire website [16]. Some (non-compliant) examples offer no purpose categories to choose from (such as Figure 2.1), while others provide an unreasonably large number of categories (see Figure 3.1). In section 3.2 we will select a balanced set of purpose categories which is not too sparse, but also does not overwhelm the user.

The heterogeneity of consent notice designs complicates the implementation of an efficient crawler. Since the structure of a cookie banner can vary greatly between websites, basing the extraction of label data on the specific CSS or HTML structure does not generalize well to other websites, even those that make use of the same CMP. Furthermore, as some CMPs do not outwardly present category labels or cookie information at all, data extraction may altogether be impossible.

Therefore, in order to efficiently gather a large dataset of cookie labels, we specify the following criteria a CMP must fulfill in order to be a suitable target for our consent label crawler:

1. *The CMP must reliably use purpose categories.* – Most consent notices prior to the introduction of the GDPR took on the form of a simple banner or popup that only inform the visitor that cookies are used, but display no further information. Unfortunately, such notices are still common, even after the introduction of the GDPR [55]. These are generally not useful for crawling, as there is no data to extract.

2. *The CMP must list cookies by category.* – Particularly for CMPs that implement the TCF, it is common for the popup to only list vendors, i.e., third party hosts that set cookies on the visitor's browser [26, 38]. Even if a CMP provides many different purposes, these may be of no use to the crawl if the popup or internal CMP implementation itself does not declare which cookies are mapped to which category.

3. *The CMP should ideally be hosted remotely.* – Due to a wide range of customization options, consent notice designs may vary greatly between websites. Some CMPs however host the required consent data on a remote domain, rather than the website directly. By directly accessing this location, we can bypass the website HTML and reliably retrieve cookie labels, regardless of the outward presentation of the consent notice.

Another consideration to make when selecting CMPs is that we require a method through which the crawler can provide full consent to the consent notice automatically. Otherwise, the notice itself will prevent the creation of cookies, as they have not been consented to yet. For this, we used the browser extension *Consent-O-Matic* [28], which automatically interacts with specific CMP dialogues in order to enforce the user's consent preferences.

Using these criteria, we analyzed 22 of the market-leading CMPs as reported by the technology trend database *BuiltWith* [8]. These were sorted by the number of occurrences in the top 1 million websites by traffic worldwide. Examining the market leaders in the area of consent management allows us to gather many potential domains to scrape data from. BuiltWith also provides a helpful set of example domains that is likely to make use of the CMP, which helped us analyse which CMPs were viable for our crawler.

However, we acknowledge that this is not an exhaustive analysis, and may potentially be biased towards domains popular in the USA. For future work, additional potential crawl targets may be retrieved and analysed from resources such as the TCF's official CMP listing [26].

To perform the analysis, we selected a sample of 5 websites that we confirm host the selected CMP, and then determined how suitable the CMP is according to our criteria. The results can be viewed in Table 3.1. Entries where checkmarks are put into brackets indicate that the criteria is only fulfilled for some occurrences, or that the CMP is generally capable of fulfilling the criteria, but that the instances found did not satisfy it in practice. Specifically in the case of *Osano*, there exists both a paid hosted service as well as a free plugin version of the consent notice, thus complicating a potential crawl.

In summary, we find that 13 of the 22 CMPs are hosted remotely, 10 out of 22 CMPs display purpose categories, and 9 of the 21 CMPs associate cookies with categories. Out of the 22 CMPs we analyzed, only 7 satisfy all three criteria that we need to perform an efficient web crawl.

We implemented a practical consent crawler that targets *Cookiebot*, *OneTrust*, and *Termly*. Note that *CookiePro* and *OptAnon* are included, as they have been acquired by OneTrust and use the same design.

3.2 Selecting Cookie Categories

Having selected a set of CMPs to crawl, we next need to define a practical set of categories that are unambiguous, can represent a user's consent preferences well, and can furthermore be understood easily by the average user of a browser extension. We decided to use the following categories, as originally proposed by the International Chamber of Commerce of the UK [27]:

1. *Strictly Necessary Cookies*
2. *Functionality Cookies*
3. *Performance/Analytics Cookies*
4. *Advertising/Tracking Cookies*

There are several reasons why we decided on these categories. The first is that they match our personal assessment of cookie purposes, as described

Table 3.1: Listing of Privacy Compliance Tools in the top 1 million sites by traffic worldwide, taken from BuiltWith [8]. Coverage is ratio of websites in the top 1 million where the CMP is used, as reported by BuiltWith. Entries that are not CMPs of some form have been removed.

Potential CMP	Coverage	Hosted?	Choices?	Has Labels?
Cookie Consent by Osano	2.25%	(✓)	(✓)	✗
Cookie Notice	1.29%	✗	(✓)	✗
OneTrust	1.17%	✓	✓	✓
OptAnon	1.08%	✓	✓	✓
Cookie Law Info	0.95%	✗	(✓)	✗
Cookiebot	0.77%	✓	✓	✓
Quantcast CMP	0.68%	✓	(✓)	✗
UK Cookie Consent	0.33%	✗	✗	✗
TrustArc Cookie Consent	0.26%	✓	(✓)	✗
WP GDPR Compliance	0.20%	✗	(✓)	✗
Moove GDPR Compliance	0.18%	✗	(✓)	✗
tarteaucitron.js	0.16%	✗	✓	✗
Usercentrics	0.16%	✓	✓	✗
CookiePro	0.15%	✓	✓	✓
Borlabs Cookie	0.12%	✗	✓	✓
EU Cookie Law	0.12%	✗	(✓)	(✓)
PrimeBox CookieBar	0.09%	✗	✗	✗
Cookie Script	0.07%	✓	✓	✓
Cookie Information	0.06%	✓	✓	✓
Termly	0.05%	✓	✓	✓
Cookie Info Script	0.05%	✓	✗	✗
Easy GDPR	0.04%	✓	(✓)	✗

in Section 2.1.1. The second is that these categories represent a stepwise increase in the potential privacy threat, in the order listed above. *Functionality* cookies do not track the user, but unlike *Strictly Necessary* cookies, they are not required for the website to operate. *Analytics* cookies serve to measure a website's performance with anonymous, single-website tracking, while *Advertising/Tracking* cookies are the most privacy-concerning, as they track visitors and collect personal data across multiple websites across the internet. Additionally, the category of *Strictly Necessary* is officially recognized by the ePrivacy Directive [21], and does not require consent.

The third major reason is that the CMPs we selected use a very similar selection of categories to suggest default purposes for cookies to their customers [13, 50]. In addition, Cookiepedia also makes use of the same ICC categories to sort cookies into purpose categories. In Chapter 6, we will use this repository as the baseline to compare the performance of our classifiers to. Using the same categories simplifies this comparison significantly.

3.3 Web Crawler Implementations

As shown in the appendix of the work by Hils et al. [25], as of January 2020 the cumulative market share of Cookiebot and OneTrust among the top 1 million websites was roughly 1%. BuiltWith [8] reports a market share of roughly 3.17% in the top 1 million websites by traffic as of February 2021.

These values indicate that a CMP crawl performed on a similar selection of domains will not find any useful data on the overwhelming majority of all websites. Therefore, to gather the desired cookie labels in a reasonable amount of time, we split the task up into two separate crawlers.

The first implements a fast, parallel approach that determines the presence of a CMP on a website using simple GET requests. This filters out the large majority of domains that do not use the selected CMP types.

The second crawler implements a slower process that retrieves the cookie and consent category information using actual browser instances. A browser instance can execute scripts, retrieve images and perform other actions which may trigger the setting of cookies, but is much slower and more memory-intensive than sending simple GET requests. Therefore, it uses the output of the former to reduce the time spent with unsuccessful crawl attempts.

We will describe these crawlers in the following subsections.

3.3.1 CMP Presence Crawler

As the name implies, the CMP presence crawler serves to detect whether a script or URL corresponding to a Consent Management Platform is present

```
Cookiebot Domains:
> https://consent.cookiebot.com/

OneTrust Domains:
> https://cdn-apac.onetrust.com
> https://cdn-ukwest.onetrust.com
> https://cdn.cookielaw.org
> https://optanon.blob.core.windows.net
> https://cookie-cdn.cookiepro.com
> https://cookiepro.blob.core.windows.net

Termly Domains:
> https://app.termly.io/
```

Figure 3.2: Domains associated with the one of the supported Consent Management Platforms. These are used for detecting CMP presence, as well as to retrieve consent label data.

on a large set of websites. It is implemented in Python 3.8, primarily using the *requests* library from the Python Standard Library to perform GET requests, and the *pebble* library to support multiple concurrent processes running at the same time.

The crawler first attempts to determine whether the given domain is reachable at all. For this, it will sequentially try both the `HTTP` and the `HTTPS` protocols, with or without the "`www.`" prefix prepended to the domain. If a connection was successful, it retrieves the HTML source of the website and performs a text search to determine if any of the domains listed in Figure 3.2 are present. Since the selected CMPs are implemented through scripts retrieved from a remote domain, this serves as a reasonably accurate indicator of whether a website utilizes one of the supported CMPs. If the connection failed or no CMP is present, the domain will be filtered out.

The crawler separates the given input domains into the following lists:

- Websites with a Cookiebot, OneTrust or Termly CMP respectively.
- Websites without any recognizable CMP.
- Websites where the connection failed.
- Crawler timeouts while parsing the site.

These can be used in a second step to retrieve the cookies with associated purpose.

To speed up the process, we run the crawl with several hundred parallel processes. Having a large number of concurrent threads works well even on consumer-grade CPUs in this setting, as most of the used time is spent waiting for responses to arrive from different hosts, thus allowing a significant performance increase beyond the number of available processors. To prevent

bot detection from denying access to websites, we use a user-agent string in the GET request header that imitates a Chromium browser instance, which are the most common types of browsers used today.

Note that we are aware of some drawbacks with this approach, which may introduce bias into the collected dataset:

- For one, the crawl may fail to accurately detect CMP presence in cases where AJAX is used, which involves loading parts of the website only after certain sections of JavaScript code has been executed. As this crawler only sends GET requests, no JavaScript code is run, and thus any CMP domains that are loaded in late will be missed. Some websites may also have landing pages that need to be passed first in order to see the consent notice, which this crawler cannot do.
- On the flipside, some websites may not actually implement the detected CMP, for example if there are leftover comments involving the domain in the HTML code. Others may have made errors in setting up the consent notice, leaving it nonfunctional.

Therefore the result will not include an exhaustive set of domains that use CMPs, and also contain some cases where no data can be extracted. We expect these occurrences to be relatively rare however.

3.3.2 Consent Label Crawler

The *consent label crawler* uses the output of the presence crawler to maximize the number of category labels and the associated cookies found, while keeping the total time required to perform the crawl low. We implemented the crawler on top of the *OpenWPM* framework, version 0.12.0 [19, 42].

OpenWPM is hereby an open-source web-privacy measurement tool suited for large-scale crawls, which compared to the web testing tool Selenium[1] allowed us to save a significant amount of development time, and provided the following advantages:

- *Error handling* – According to Englehardt et al., Selenium has not been designed to support large-scale crawls, and the browser may frequently run into internal errors that need to be recovered from [19]. OpenWPM has built-in crash handlers that can restore the browser profile whenever an instance of the browser fails.
- *Built-in parallelism* – OpenWPM supports running multiple Selenium browser instances to crawl different websites simultaneously. This combined with the built-in database aggregator allows us to greatly improve the performance of the crawl.

[1] https://www.selenium.dev/

- *Full browser instrumentation* – Nearly every aspect of the browser is instrumented in OpenWPM, including cookies. By using OpenWPM we can collect cookie data and consent labels at the same time.
- *Database aggregator* – OpenWPM uses an SQLite database with an aggregator that serializes concurrent storage commands. We use this database to store the collected labels for each cookie, and to perform additional analyses on the collected data.

In addition to using the OpenWPM framework, we have made several additional considerations in the setup of the consent crawler. These are:

- *Geolocation difference* – Prior work by Dabrowski et al. [15] has indicated that many websites, particularly those among the Alexa top 1000, engage in some form of geographic discrimination. The GDPR stipulates that its rules apply to all hosts that offer their services to EU visitors [47]. Therefore many websites distinguish between EU and non-EU users [18] by selectively displaying their consent notices depending on what geographic location the visitor is connecting from. As this work was conducted from within Switzerland, it was necessary to perform the crawl while connected to a VPN located in a EU member state. This ensured that consent notices were properly displayed and accessible during all stages of the crawl.
- *Providing consent* – While we need the consent notice to gather purpose labels, it also prevents the crawler from retrieving all the cookie data until consent is given. To solve this problem, we installed the Consent-O-Matic extension [28] into the browser profile, configured to automatically provide consent to all purpose categories.
- *Browser settings* – To ensure that no cookies are inadvertently blocked, we disable all browser-integrated tracking protection, including the *Do Not Track* header [48], all browser-internal cookie-blocking mechanisms, and ensure that image loading is enabled.

The crawling process then works as follows: URLs from the input sample are sent to the browser instances in a first-come, first-serve manner. Each instance first connects to the website's landing page, and if successful, tries to determine which CMP is present on the website. Once a CMP type is found, the cookie label extraction process for that type is then performed. If no CMP is detected, or the extraction of labels has failed, the process is aborted and the next URL is crawled.

The next step after a successful label extraction is to browse the website by detecting subpage links present on the landing page, and then accessing a prespecified number of them. For each subpage, the crawler scrolls down to the bottom of the page and performs random cursor movements to trigger

```
SELECT DISTINCT *
FROM consent_data c
JOIN javascript_cookies j ON c.visit_id == j.visit_id
                          and c.name == j.name
WHERE j.record_type <> "deleted"
ORDER BY j.visit_id, j.name, c.time_stamp ASC;
```

Figure 3.3: SQL statement used to join the cookies and consent tables.

the setting of cookies. Previous work by Urban et al. [53] has shown that accessing subpaths of the crawled website can increase the amount of cookies gathered by up to 36%. In the interest of crawl performance, we however only access 5 randomly chosen links to subpages for each crawled domain.

In Appendix A, we provide detailed descriptions on how the consent labels, as well as other related information, are retrieved for each supported CMP. This includes Cookiebot, OneTrust and Termly.

3.4 Training Dataset Construction

The output of the consent crawl is an SQLite database containing the collected cookies, the retrieved consent labels and descriptions, as well as debug information on the success and failure states of the crawl. The main content is split into two database tables:

- `javascript_cookies` – SQL table that contains all actual browser cookies encountered during the browsing phase of the crawl.
- `consent_data` – SQL table that contains the information retrieved from the CMP, including cookie name, host, purpose description and label.

With these tables, we need just one more step in order to retrieve a valid training dataset. Namely, we need to match each actual observed cookie with its corresponding declaration from the consent notice. To do so, several steps of preprocessing are required.

First, the tables need to be joined, like shown in the SQL statement listed in Figure 3.3. This matches all actual cookie names with the names found in cookie declarations, if and only if the declaration was found on the same website as the actual cookie. The latter condition is needed because the same third-party cookie may be found on different websites, and may furthermore be assigned different labels depending on the domain it was encountered on. To deal with such contradictions, we identify each cookie with the domain that the web crawler visited at the time of collection. Thus, each third-party cookie becomes unique, and is treated as a separate training sample.

The "`record_type`" column in the database indicates whether a cookie was created, updated or deleted. We are only interested in newly created cookies and updates to existing cookies, as the deletion of a cookie normally only occurs upon being replaced by an update. Correspondingly, a cookie update occurs whenever the browser attempts to create a cookie that already exists. In this case, its content, expiry date as well as all associated metadata may change. Cookies may be updated multiple times over the course of a crawl, and all the changed data can be considered as features for training.

After executing the SQL statement, we utilize a separate Python script to perform additional steps of pre-processing on the result of the query:

- First, we aggregate all updates for a unique cookie in an array, such that each cookie identifier represents a single training data entry.
- Second, we need to filter out records where the origin of the cookie is not listed in the declaration. Because the format of the cookie's actual domain may slightly differ from the one found in the declaration, we first need to transform both strings into a uniform representation before a comparison can be made.
- Third, all cookies that are unclassified, and furthermore all cookies where the category could not be recognized, have to be filtered out from the dataset, as they are not useful for training.
- Finally, we also remove specific *"consent cookies"* from the dataset. This includes the "`CookieConsent`" cookie from Cookiebot, as well as the "`OptAnonConsent`" cookie from OneTrust. Both store the user's choices after a consent notice has been interacted with. This is done to prevent the classifier from becoming biased. Due to the way domains are selected for crawling, these cookies appear on almost all of the targeted domains. Conversely, they never appear on domains that are not part of the target dataset. Combined with the fact that they always have the same internal structure and label, such cookies represent a clear form of training bias that must be removed.

After the data is processed, the Python script outputs a JSON document that contains the training data samples. We decided to use JSON format so that the content can easily be transferred and reused across different codebases. This was necessary as we ultimately needed to implement the feature engineering process not just in Python, but in JavaScript through NodeJS as well. This is a decision we will elaborate on in Chapter 5.

Our final dataset contains 309 472 cookies, collected from a selection of 26 403 unique domains. An exact breakdown of the crawler performance, including the selection of domains and statistics on the success and failure rate, as well as a detailed analysis on the collected cookies and consent labels will be provided in the evaluation in Chapter 6.

Chapter 4

Cookie Consent Classifier

Machine learning on the domain of cookies has so far not been explored in scientific literature. To train a classifier, we first need to determine what patterns and properties exist in cookies, and then define a feature extraction process to transform these contents into a numerical representation. We give an overview of cookie data next, and then show how to transform it into numerical embeddings. Finally, we present the applied classifier approaches with some background information.

4.1 Cookie Properties

In Section 2.1, we provided a high-level description of browser cookies and what they are used for in practice. Here, we will provide a more detailed look on the individual components for the purpose of feature engineering. Most of this information is sourced from Mozilla's online documentation on browser cookies [41]. A cookie consists of the following data:

- *Name, Domain and Path* – These three components serve as the key of the cookie, which uniquely identifies it across the web.
  - The *name* can only consist of US-ASCII characters. Whitespaces, control characters and separators such as "`()<>;,`" can also not be used in the name of a cookie.
  - The *domain* identifies the origin that originally set the cookie, and may begin with a leading period character, for instance like in "`.example.com`". If present, this period indicates that the cookie may be used by all subdomains as well.
  - The *path* determines which subdirectories of the domain can access the cookie. The default if unspecified is `"/"`, which indicates that all subdirectories are granted access.

- *Value* – Represents the payload of the cookie. A cookie can only contain up to 4 kilobyte of ASCII text data. If Unicode characters are used, the contained text will be URL encoded.[1] Like the name field, it also cannot contain whitespaces, control characters, semicolons or commas.
- *Expiry* – Specifies the maximum lifetime of a cookie, in the form of a fixed date or in seconds. We distinguish between two cases:
  - If the expiry is specified, the cookie is referred to as *persistent*. The cookie is removed after the local date has passed the expiry.
  - If the value is not specified, then the cookie is referred to as a *session* cookie. It lasts until the current browser session ends.
- *Host-Only flag* – If true, the browser does not allow any subdomains to read or write to the cookie. Equivalent to leading period in domain.
- *HTTP-only flag* – If true, the cookie may not be read or written to by JavaScript code. Note that browser extensions still have access to the cookie through a specialized API.
- *Secure flag* – Indicates that the cookie may only be sent over secure connections, i.e., HTTPS. They are however not stored encrypted.
- *SameSite flag* – Defines the policy for when the browser makes a cross-origin request, i.e., a request for a domain that is not a subdomain of the current. This is a categorical feature with 3 possible values:
  - *"None"* – The cookie is always sent with a request to the domain, no matter the origin.
  - *"Lax"* – The cookie may not be sent with implicit cross-origin requests, for example when third-party images or frames are loaded when browsing a domain. They are only transmitted if the user explicitly clicks a link to browse to a remote domain.
  - *"Strict"* – Like "Lax", but the cookie is also not sent when the client explicitly browses to a cross-origin domain. Only once a subdomain is requested is the cookie finally transmitted.

Also note that a cookie's expiry, content and flags are not constant, and can be altered by JavaScript or HTTP requests. Each event that could potentially change the contents of a cookie will be henceforth referred to as an *update*, even if no actual data was altered. A concrete example for this are cookies that used to store the consent preferences of the user. At first, the cookie will be empty. After selecting the consent preferences however, the cookie's contents will be updated to reflect those settings.

[1] Encoded by the character % followed by two hexadecimal digits.

4.1.1 Third-Party Status

Cookies are commonly distinguished as *"first-party"* and *"third-party"*. The former are cookies that have been created by the same domain that the user is directly browsing. The latter are cookies where the origin does not match the domain in the browser address bar at the time of creation.

Third-party cookies are created through implicit requests to external resources, such as images, or through scripts running on an embedded frame, such as a widget stemming from a social media website. They are commonly used to track the user, but may also serve to enable essential website functionality. An example for this is *Single Sign-On (SSO)* authentication, where a single third-party cookie ensures that a user is logged in on multiple different domains. Hence a simplistic approach that simply blocks all third-party cookies may potentially break website functionality. In addition, both first- and third-party cookies may be used to collect personal information.

Note that in early 2020, Google has announced their intent to remove support for third-party cookies from Chrome. They claim that they intend to replace the use of third-party cookies with privacy-preserving alternatives. An example for this is the recent *"Federated Learning of Cohorts (FLoC)"* API which is supposed to replace traditional tracking approaches for the purpose of personalized advertising [51].

Historically, due to the high market share of Chromium-based browsers [49], decisions made by Google usually lead to other browser vendors following suit. As such, this may soon lead to significant, observable changes in how cookies are used on the Web.

4.1.2 Content of a Cookie

In addition to a cookie's innate values and properties, it is also important to determine what kinds of data is usually stored in a cookie's content. To determine what features could be extracted, we manually analyzed the cookie dataset to determine common patterns.

We focused primarily on features that are generic and can be found in any cookie regardless of origin. In this manner, the classifier can better generalize to instances found in the wild. Furthermore, such features are unlikely to change or disappear over time. Some example features found in the cookie content are:

- Hexadecimal numbers
- Dates and timestamps
- Separator characters

While it would be possible to condition the classifier on vendor-specific content patterns, this would likely be redundant, as the cookie domain already allows the classifier to distinguish cookie category based on the origin.

One property specific to tracking cookies is that the same identifier can be seen across multiple sites, if and only if the browser state is retained between visiting different domains. This cannot be recognized in the current classifier model, as capturing cross-website relations is not possible when classifying each cookie in isolation. Moreover, such identifiers are unique to the user, and thus cannot help in the conditioning of a classifier. Instead, we use different measurements, such as the entropy, to help distinguish which cookies are likely to store unique identifiers used to track visitors. See also Appendix B.4 for more information on how this is done.

4.2 Feature Engineering

The contents of a cookie include ASCII text, categorical data, numerical values, dates, and boolean flags. As the types of its content are greatly varied, we transform these contents into uniformly-sized vector representations of numerical data. This can then be used as input to a large range of classifiers.

To allow the classifier to be applicable to any cookie, we can only perform the training on data that is directly available inside a browser. As a result, information such as purpose description obtained through a Consent Management Platform cannot be used in the following feature extraction.

In total, we determined a total of 52 feature engineering steps which represent the properties of a cookie as a real-valued vector. These steps are split into three separate groups:

1. The first consists of what we denote as *"per-cookie features"*, which are extracted exactly once per unique training sample. This includes all attributes of a cookie that remain constant and are never changed in any update, including *name*, *domain* and *path*, as well as averages and standard deviations over all cookie updates. The full list is shown in Table 4.1. Note that a *one-hot vector* is a sparse vector in which exactly one entry is set to 1, while all others are 0. These vectors are used to represent categorical features.

2. The second group consists of *per-update features*, which are extracted exactly once per update for each cookie, up to a predefined limit. Updates are hereby considered in the order they have been encountered, sorted by timestamp. The features are derived from values which may change in an update, including the flags, the expiry and the cookie's content. The complete list of per-update features is shown in Table 4.2.

Table 4.1: Per-Cookie Features Overview: All features that are extracted once per unique cookie. Entries marked with a * could cause issues when used within the context of a browser extension.

Feature Name	Size	Description
Top Names	100	One-hot vector of the most common exact names.
Top Domains	100	One-hot vector of the most common domains.
Pattern Names	50	One-hot vector of the most common name patterns.
Name Tokens	100	Binary indicator of English tokens in the name.
IAB Vendor	1	Binary Indicator, true if domain is an IAB Vendor.
Domain Period	1	Indicates whether the domain starts with a period.
Third-Party *	1	Whether the cookie originates from a third-party.
Non-Root Path	1	Whether the cookie path is not the root path ($\neq$ "/").
Update Count *	1	Total number of updates encountered for this cookie.
Host-Only Flag	1	Whether the "host-only" flag is set.
HTTP-Only Changed	1	Whether the "HTTP-only" flag changed in any update.
"Secure" Changed	1	Whether the "secure" flag changed in any update.
"Same-Site" Changed	1	Whether the "same-site" entry changed in any update.
"Expiry" Changed	1	Whether the expiry changed by >1 day between updates.
Content Changed	1	Whether the cookie content changed between updates.
Levenshtein Total	2	Mean and Std.Dev. of Levenshtein dist. between updates.
Difflib Total	2	Mean and Std.Dev. of Difflib similarity between updates.
Length Total	2	Mean and Std.Dev. of the cookie value length in bytes.
Compressed Total	2	Mean and Std.Dev. of the compressed cookie value length.
Entropy Total	2	Mean and Std.Dev. of the Shannon Entropy of values.

To keep the resulting feature vector at a fixed size, the number of updates is constrained to a constant factor, with any additional updates being truncated from the front. If there exists less updates than required, the remaining values will be padded by zeros. To distinguish boolean flags from missing entries, "`true`" is represented by 1, while "`false`" is represented by -1.

3. The third group consists of the *per-diff features*, which consider the changes that may occur between two updates of a cookie. Each update is sorted by timestamp, such that the feature extraction may represent changes that occurred over time. Like with per-update features, the number of updates considered for this is constrained to a constant factor, with missing entries being padded through zeros. The feature set for this group is presented in Table 4.3

Note that some of the features are not fully supported in the context of a browser extension. This will be explained further in Section 5.3.1. Moreover, a detailed description of the more elaborate features shown in the tables, along with some additional arguments as to why the chosen features are useful, can be found in Appendix B.

Table 4.2: Per-Update Feature Overview: All features that are extracted once per cookie update. Number of updates used for extraction can be specified separately.

Feature Name	Size	Description
"HTTP-Only" Flag	1	Binary indicator of whether the "http-only" flag is set.
"Secure" Flag	1	Binary indicator of whether the "secure" flag is set.
"Session" Flag	1	Whether the cookie is a session cookie or not.
"Same-Site" Flag	3	One-hot vector, whether "None", "Lax" or "Strict" is set.
Expiration Time	1	Ordinal feature, contains the expiration time in seconds.
Expiration Intervals	8	Interval checks on expiration time, e.g., >1 day, <1 week.
Content Length	1	Total size of the cookie's value in bytes.
Compressed Length	2	Size and reduction of cookie value after *zlib* compression.
Shannon Entropy	1	Shannon entropy of the cookie update's value.
URL Encoding	1	Indicates whether the cookie content is URL encoded.
Base64 Encoding	1	Indicates that the content is potentially Base64 encoded.
Delimiter Separated	1	Number of delimiter separations (i.e. CSV data entries).
Period Separated	1	Number of separations of the content by periods.
Dash Separated	1	Number of separations of the content by dashes.
Contains JSON	1	Whether the content contains a JSON object.
Content Terms	50	Binary indicator of English tokens in the content.
CSV Contents	5	Try to split as CSV and detect content types within.
JS Contents	11	Try to split as JSON and detect content types within.
Numerical Content	1	Whether the content consists entirely of numbers from 0-9.
Hexadecimal Content	1	Whether the content represents a hexadecimal number.
Alphabetical Content	1	Whether the content is entirely alphabetical.
Identifier Content	1	Whether the content is a valid code identifier.
All Uppercase	1	Whether the cookie content has all uppercase letters.
All Lowercase	1	Whether the cookie content has all lowercase letters.
Empty Content	1	Whether the content of the cookie is empty.
Boolean Content	1	Whether the cookie content is a boolean of some form.
Locale Content	1	Check if content includes a country or currency identifier.
Timestamp Content	1	Check for a UNIX timestamp in the cookie content.
Date Content	1	Check if content contains a date term or identifier.
UUID Content	6	Check if content contains a valid UUID, and which variant.
URL Content	1	Check if the content contains a URL of some form.

Table 4.3: Per-Difference Features Overview: All features that are extracted as comparisons between two contiguous updates, sorted by timestamp.

Feature Name	Size	Description
Expiry Difference	1	Expiration time difference in seconds between two updates.
"Difflib" Similarity	1	Similarity ratio between cookies, as measured by "difflib".
Levenshtein Distance	1	Levenshtein distance between two cookie updates.

4.3 Classifier Description

In this work, we try to apply existing classifier approaches to the domain of browser cookies. Our goal is to utilize hidden patterns in the data to distinguish what purpose a cookie is used for. As this will serve the enforcement of cookie consent preferences on the web, the number of inaccurate predictions should ideally be very low.

To achieve high performance, we decided to use ensemble tree methods to perform the classification. We primarily focused on applying the XGBoost algorithm [10], as it is among the most powerful classification technologies in use today. In particular, it is suited for training on sparse feature matrices with missing data, which our dataset includes because of the fixed number of cookie updates per training sample.

XGBoost is a well-established algorithm for training regressors and classifiers, which has been used widely to achieve top-performing results in machine-learning challenges. Both training the model and prediction of new labels is very fast, and scales well with thousands of features, even without the use of high-performance hardware. This made it an ideal first choice to tackle our goal.

In addition to XGBoost, we also apply two other ensemble tree algorithms to the task, those being LightGBM [32] and CatBoost [17], which have specific advantages. In the following, we will provide a high-level explanation of the necessary machine learning topics, as well as a short description of the individual algorithms we applied.

4.3.1 Machine Learning Background

In this section we recall the necessary background information on the machine learning concepts used in creating the cookie consent classifier.

Tree ensemble classifiers, such as XGBoost, are trained through supervised learning. Given a set of training samples $X = (x_1, \ldots, x_N)$ with associated labels $Y = (y_1, \ldots, y_N)$, the classifier is tasked with learning a representation of the data such that the predicted labels $\hat{Y}$ match the ground truth Y as closely as possible. A single prediction $\hat{y}_i$ is hereby derived from some representation of the corresponding training sample x_i. An example for this is a linear model, where the prediction for a single sample x_i is computed as $\hat{y}_i = \sum_j \theta_j x_{ij}$, where θ_j are the model parameters.

For training the model, we need a cost function $L(X, Y; \theta)$ that quantifies how well the parameters θ fit the training dataset X, Y. In all the classifiers we apply, we use the multi-class cross-entropy loss for L. Using the class probabilities p_{ik}, which are the predicted probabilities for a sample x_i to

belong to class k, and a binary indicator z_{ik}, we define the multi-class cross-entropy as follows:

$$L(X, Y; \theta) = -\sum_i \sum_k z_{ik} \log(p_{ik}(\theta))$$

Note that $z_{ik} = 1$ if k is the ground truth label for sample x_i, and 0 otherwise. The class probabilities p_{ik} are computed through the application of a softmax transformation to the scores output by the model when run on a given sample x_i. In simple terms, the relative ratio of the score for class k compared to the total score over all classes produces p_{ik}.

Classifiers may be prone to *overfitting*, which occurs when the classifier is excessively fit to the training set and fails to generalize well to newly observed examples. To prevent overfitting, many models add to $L(X, Y; \theta)$ one or more regularization functions $\Omega(\theta)$. In the case of tree ensemble models, this regularizer aims to reduce the complexity of the forest of trees.

The combined objective of the classifier for training is thus to minimize the cost function $L(X, Y; \theta)$ as well as the regularizers $\Omega_j(\theta)$:

$$\min(\sum_i L(x_i, y_{ik}; \theta) + \sum_j \Omega_j(\theta))$$

4.3.2 Tree Ensemble Classifiers

The model used in Gradient Tree Boosting is an ensemble (or forest) of classification and regression trees (*CART*). These trees are constructed similarly to ordinary decision trees in that each inner node represents a boolean decision operating on a single feature of the input sample.

Unlike a decision tree, the leaves in a CART do not represent a fixed decision, but rather a positive or negative score guiding the decision. A single tree is usually not strong enough to reach a decision, hence the training constructs multiple trees. Each tree operates on a different set of features, and the final decision is based on the total sum of scores obtained from all trees after an input sample is applied. In the case of multi-class classification, each class receives a forest of trees, and the class with the best score is typically chosen, depending on the decision rule.

It is too computationally expensive to enumerate all possible combinations of trees in a single step. Therefore, trees are created in an additive manner, extending the tree with additional nodes until no improvement is found. Then the tree is finalized, and a new one is added to the forest. Trees are extended by splitting a leaf node into a new binary decision rule with additional leaf nodes.

Only trees that further optimize the objective are added, and only those nodes are split where the gain in score outweighs the gain in complexity, as

restricted by the regularization functions. The structure of the ensemble, the features used for the decisions, as well as the resulting leaf weights are all parameters of the model that are learned during training.

The resulting model of the tree ensemble is thus the forest of trees itself. This model can easily be extracted, stored in a text format, and then reimplemented in another setting without requiring any complex numerical computations. We use this to implement the predictor in the browser extension setting using only plain JavaScript.

For our task, we apply three different classifier approaches, each being a tree-ensemble model with different properties. These are:

- XGBoost:[2] A sparsity-aware gradient tree boosting algorithm developed by Chen and Guestrin [10]. It uses a pre-sorted and histogram-based algorithm for computing the best split in each node. Its main advantage compared to tree boosting techniques that came before it is its scalability and performance. As such it became one of the most recognized classifier approaches in use today.
- LightGBM:[3] A gradient-boosting decision tree algorithm designed by Ke et al. [32]. It makes use of two techniques called *Gradient-based One-Side Sampling (GOSS)* and *Exclusive Feature Bounding (EFB)* to train the classifier. GOSS hereby allows the training process to focus only on data samples with steep gradients, using the remainder for information gain. This massively speeds up the training process, which is its main draw over XGBoost.
- CatBoost:[4] This algorithm, created by Prokhorenkova et al. [17], introduces a technique called "*ordered boosting*" which counters a statistical shortcoming of XGBoost and LightGBM, namely in regards to target leakage. The name itself stands for "categorical boosting", owing to the novel method in which categorical features are handled by the algorithm. The authors report that their approach outperforms both XGBoost and LightGBM in classification performance.

4.4 Additional Considerations

In addition to the choice of classifier approach, we also needed to consider some properties specific to the task that we train the predictor for. These will be discussed below.

[2] XGBoost source: https://github.com/dmlc/xgboost
[3] LightGBM source: https://github.com/Microsoft/LightGBM
[4] CatBoost source: https://github.com/catboost

4.4.1 Class Imbalance

In our dataset, there exists a significant imbalance in the number of cookies in each class. Such imbalance may be inherent to the setting of cookie categories. In Section 6.1.3, we will see that cookies belonging to the class of "Advertising" cookies are significantly over-represented, and make up 55.94% of all declarations. Conversely, cookies that belong to the "Functionality" class are only represented by 9.90% of declarations.

Such class imbalance causes problems not only for achieving a high-quality prediction, but also for properly assessing the quality of the classifier as a whole. With "Advertising/Tracking" being this over-represented, the resulting classifier will likely be biased towards achieving a higher accuracy in correctly identifying those types of cookies, and have reduced accuracy for all other instances. The same problem is then observable in the evaluation, as the total dataset accuracy may not properly represent the real performance of the classifier.

To reduce the effect of class imbalance in training, we multiply the loss for each sample with the inverse class weight, corresponding to the class label. The inverse class weight for a class j is computed as:

$$w_j = \frac{\text{total number of samples}}{\text{number of samples in class } j}$$

This approach is supposed to counteract the effect of imbalance, and give under-represented classes a higher priority. It is however not perfect, and the effects of the imbalance will still be visible in the evaluation.

4.4.2 Impact of Misclassifications

Given an input sample x_i, the resulting predictor model produces an array of probabilities which indicate how likely the given sample is to belong to one of the four cookie consent purposes described in Section 3.2.

The question is, given these probabilities, how will the predictor produce a discrete label for the given input cookie? The simplest approach would be to always use the label with the maximum class probability as the corresponding prediction. Thus, if for example the probability for class "necessary" for a given input sample is 40%, and the probability for the remaining three classes is 20%, it will predict "necessary" as the discrete label.

This simple decision may not be enough. We need to also take into account that in the setting of cookie consent and user privacy, not every type of error has the same impact. Consider the following cases:

- A cookie with ground truth "Advertising" is misclassified as "Strictly Necessary". In this case, the privacy of the user is potentially endangered as the cookie will simply pass through and be accepted, even if the user rejected advertising cookies.
- A cookie with ground truth "Strictly Necessary" is misclassified as "Advertising". This will break website functionality if the user intends to block advertising cookies. As a result, the user experience is negatively affected.
- A "Functionality" cookie is classified as "Necessary", or vice-versa. While undesirable, this has a reduced user impact compared to the other cases, as the categories are more similar and have less privacy impact. We also expect it is unlikely that users will want to block functionality cookies.

Generally, the chosen purpose classes can be ordered from least to most privacy-concerning, and from least to most service-critical. A misclassification is more severe the greater the distance between the true and the predicted class is on these scales.

With noisy labels and an imperfect predictor, the goals of ensuring both website functionality and ensuring full privacy are hard to fulfill at the same time. We attempt to find an appropriate balance between the two opposing goals. To achieve this in practice, we can apply the concept of Bayesian Decision Theory to the predicted class probabilities [3].

To transform the probabilistic classifier c into a discrete classifier $\hat{c}$ over our chosen purpose labels, we need to define a loss function $L(y, \hat{y})$ which quantifies the cost of misclassifying a potential ground truth y as label $\hat{y}$. The discrete classifier $\hat{c}$ is then defined as follows:

$$\hat{c}(x) = \arg\min_{y}(\sum_{y} p(y \mid x)L(y, \hat{y}))$$

Where $p(y \mid x)$ is the predicted probability for label y given input sample x.

The cost function $L(y, \hat{y})$ for our set of purpose categories can hereby be defined as a 4x4 matrix. The columns represent the predicted class, while the rows represent the ground truth. Furthermore, the classes are ordered such that index 0 corresponds to "Strictly Necessary", index 1 corresponds to "Functionality", index 2 corresponds to "Analytics" and index 3 corresponds to the "Advertising" class. The entry L_{ij} then represents the cost factor for the case where ground truth i is predicted as label j. For the correct prediction $i = j$, it makes sense to set the cost L_{ii} to zero.

The argmax predictor, where each type of misclassification is given the same weight, would thus take the form of the following symmetric matrix:

$$L_{\arg\max} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

The following symmetric cost matrix L_1 applies an increased penalty to classes that are "further away" in terms of the scale of "privacy-infringing" and "service-criticality":

$$L_1 = \begin{bmatrix} 0 & 0.25 & 0.75 & 1 \\ 0.25 & 0 & 0.5 & 0.75 \\ 0.75 & 0.5 & 0 & 0.5 \\ 1 & 0.75 & 0.5 & 0 \end{bmatrix}$$

Matrix L_2 represents a cost function which places increased focus on user privacy, in that it assigns twice the cost to misclassifications in the direction of "privacy infringement" than "service criticality":

$$L_2 = \begin{bmatrix} 0 & 0.5 & 0.5 & 0.5 \\ 1 & 0 & 0.5 & 0.5 \\ 1 & 1 & 0 & 0.5 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

The inverse can be obtained by simply obtaining the transpose of L_2.

The user is free to select which approach suits them best. For the evaluation, we selected the simple argmax cost function.

Chapter 5

Browser Extension Design

In this chapter, we present the design of CookieBlock, an extension for Chromium-based browsers and Firefox, which can automatically classify cookies found in the wild and accept or reject them based on the user's preferred privacy policy.

It is based on the feature extraction and classifier described in Chapter 4, which uses training data collected via the crawlers described in Chapter 3.

5.1 Overview

CookieBlock is built using the cross-browser WebExtensions API.[1] The goal we set out to achieve was to create an extension that offers the following functionality:

1. Automatic classification of cookies into distinct purpose categories.
2. Enforcement of a user's privacy policy by filtering cookie categories that have been rejected by the user.
3. Allowing the user to choose which consent categories to block, with an informative description being provided for each category.
4. Allowing the user to define exceptions for certain websites, such that all cookies that arrive through that website are accepted.

With the exception of point 4, which could only be partially implemented, we were able to achieve all the target functionality. Unfortunately, due to technical limitations, we are only able to allow simple exemption of first-party cookies, with third-party cookies requiring a separate exception. A potential approximate solution would be to permit all cookies if the currently active tab matches an exception, but this is as of yet unimplemented.

[1] https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions

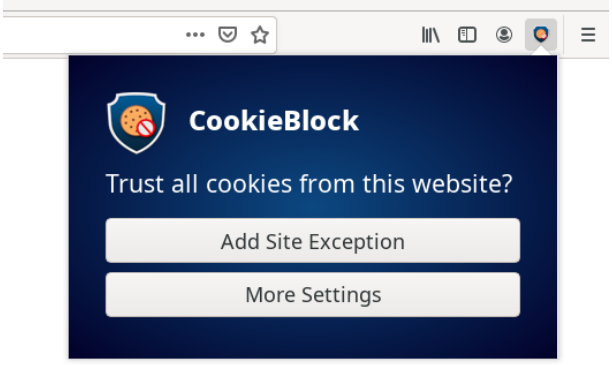

Figure 5.1: The CookieBlock popup, accessed through an icon next to the address bar. It allows the user to add exceptions for the currently active tab.

The extension consists of the following main components:

- A *background script* which intercepts cookies as they arrive, and enforces the user policy. It extracts features on-the-fly, performs the class prediction and decides whether a cookie should be deleted.
- A *first-time setup*, through which users can initialize their preferred consent choices.
- A *settings page*, which allows the user to alter their preferences, add website exceptions and view statistics on the collected data.
- A *popup*, accessible through a browser bar icon, which displays a button to add the current domain to the global website exceptions, and a button to access the settings page.

In the following, we present each part of the interface, describe the background script and elaborate on how the feature extraction and predictor were ported to JavaScript.

5.2 Extension Interface

The extension interface includes a *settings page*, a top bar icon with *popup* and a *first-time setup* that is shown when the extension is installed. The current design of the interface is based on the HTML and CSS stylesheet of the Consent-O-Matic extension, which is released under the MIT license. This may be subject to change for future versions.

The *extension popup* is shown in Figure 5.1. It allows users to open the settings page, and to add the current website as a global exception to the cookie filter. The intention behind the exception button is to allow the user to disable the policy enforcement for the current site, so that potentially broken

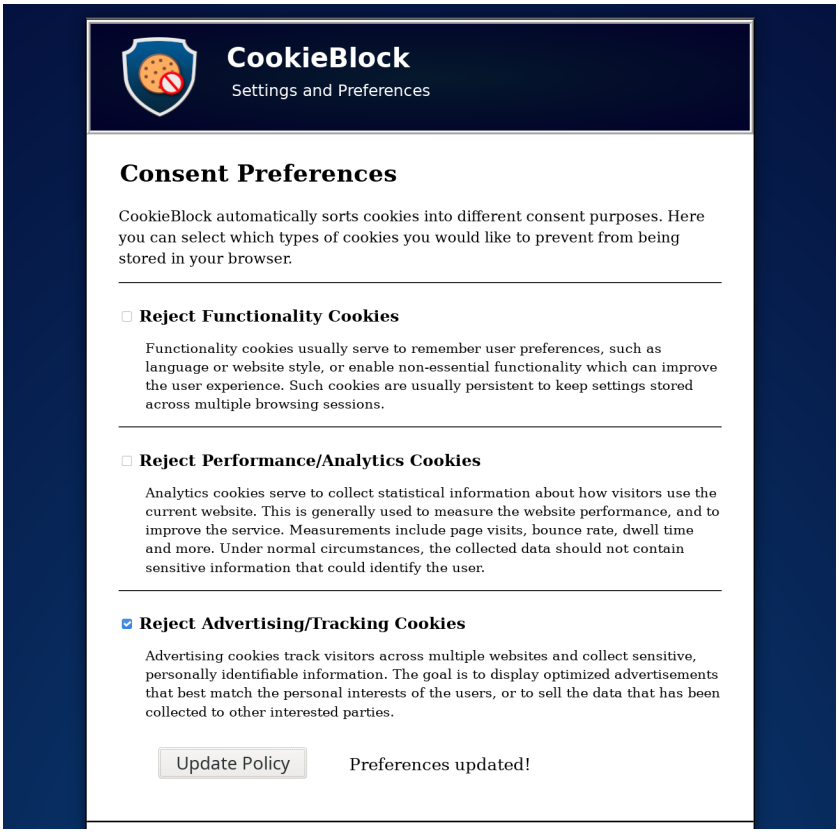

Figure 5.2: The CookieBlock settings page, accessed through the popup. It allows the user set their consent choices for all websites, and add individual exceptions. The first-time setup page is structured in a similar manner to the settings and is hence not shown here.

functionality can be re-enabled without altering the extension's behavior on other domains. At present, because the WebExtensions API only allows viewing cookies in isolation without any surrounding context, only first-party cookies are permitted. This is done by comparing the host field of the cookie to the list of exceptions. A potential solution would be to assume the domain of the currently active tab to be the origin for all cookies that are received. However, this assumption may be violated if multiple tabs are opened at the same time, or if a tab is opened in the background.

The design of the *settings page* is presented in Figure 5.2. Its primary function is to allow the user to reject cookies that are used for the purposes of *"Functionality"*, *"Analytics"* and *"Advertising"*, which we previously described in Section 3.2. Note that we explicitly exclude *"Strictly Necessary"* from this selection as it makes little sense for a user to reject cookies that enable essential website services.

In addition to global exceptions, the settings page also allows the user to

define exceptions for specific classes. For example, it is possible for the user to allow "`example.com`" to create cookies of the type "Analytics" without permitting the website to also set cookies used for advertising.

Finally we also include a *first-time setup*. This is displayed immediately after the extension is first installed, and allows the user to define their consent choices for the first time. Its design is very similar to the settings page shown in Figure 5.2, hence we will not display it here. Upon accepting the chosen settings, the extension runs a prediction on all cookies currently stored in the browser, removing the ones that belong to rejected categories.

5.3 Background Process

When the extension is launched, the background script initializes a listener object that intercepts cookies as they are created, updated and deleted. Cookie removals are ignored by the extension, while cookies that are created or updated automatically trigger the extension's classification and policy enforcement mechanism. It proceeds as follows:

- First, the extension records the data retrieved from the cookie in the "`chrome.local`" extension API.[2] This storage contains all previously encountered cookies, indexed by name, domain and path. This is done so that cookies that have been removed from the browser's storage can still be recognized if they are reconstructed by a website. This data is kept entirely local, and is never sent to any party outside the extension.
- Whenever a cookie is created, the extension initializes a queue of update slots for that cookie. This queue can hold up to 10 cookie updates in total, in the order in which they have been encountered. If the limit is exceeded, the oldest update will be evicted. See also Section 4.1 for an explanation of what a cookie update entails.
- After the cookie has been recorded, the extension checks if the cookie's domain is part of any global exceptions. If so, the classification is skipped entirely, else we continue with the next step.
- Next, the features for the current cookie are extracted. The features are output in the form of a JavaScript object representing a sparsely indexed vector of numerical data.
- The predictor is implemented in the form of a recursive function that traverses each CART tree in each of the four tree ensembles, one corresponding to each class (see Section 4.3.2). This produces the score for each category, from which the discrete label for that cookie is derived.

[2] https://developer.chrome.com/docs/extensions/reference/storage/

- Finally, using the predicted label, the extension decides whether the cookie needs to be filtered. First, if the domain of the cookie has a user-defined exception for that specific label type, the cookie is kept. Otherwise, the extension checks the user's consent policy, and if the consent category is rejected, the cookie is deleted.

Note that this process technically leaves a very small period of time between creation and deletion where the cookie can exist inside the browser. We have not yet evaluated whether this small time interval could be exploited by websites to still utilize the cookie to track users and collect personal data. If this were the case, direct browser support would be necessary to prevent the creation of cookie in the first place.

Currently, we continuously reclassify cookies whenever they appear, in case the addition of additional updates changes the prediction. This may however make it possible for websites to slow down the browser by attempting to continuously refresh cookies. Extensive evaluation will be needed to identify whether this is a problem in practice.

5.3.1 Online Feature Extraction

In order to support cookie classification in the extension, it was necessary to port the feature extraction from Python to JavaScript. To do so, we were required to make a number of changes to the process, and consider certain aspects that were previously not of importance in the offline setting. We will discuss these topics in the following subsections.

Speed of the Feature Extraction

In an offline setting, the speed at which a script can extract features from individual training data entries is not a big concern, as there is no time-pressure to get the cookie data ready for prediction. This changes in the transition to the browser extension. As the classification needs to occur on-the-fly, it is important to ensure that the feature extraction is fast enough so as not to slow down the browser and have a negative impact on the user experience.

While most of the transformations we apply to cookies are fast enough not to disturb the user experience, there are some steps that may become slow depending on the cookie content. One example for this are measurements that compute the similarity of two cookies, including the Levenshtein distance. This algorithm has a computational complexity in the order of $O(N^2)$, where N is the length of the string, thus long cookie strings will lead to a potentially slow feature extraction. Fortunately, most cookies contain very small content strings.

Update Count

The cookies obtained through the crawler described in Section 3.3.2 have some artificial properties. The most problematic one, which the classifier may condition itself on, is the overall number of updates of a cookie.

Because the crawler has a limited time interval during which it can gather cookies from a website, the average number of updates for a cookie will look rather stable. In fact, as we will see in Section 6.1.3, the number of updates obtained through this can differ depending on assigned consent class. As such, this may be a feature that the classifier can condition itself on.

The problem here arises in the online context of a browser extension. Unlike the crawler, the user does not have a limited amount of time to browse a website. Updates may accumulate over time, and quickly exceed the numbers seen by the web crawler. Thus while the update count can be used to predict labels for a fixed dataset, it is not possible to use in an online setting where the number of updates increases monotonically.

Per-Update Features

A similar problem as for the update count arises when one extracts features for more than one update per cookie. Specifically, the number of per-update features that are used in training is fixed, with zero padding if there is missing data. Here, it is possible that the classifier may condition itself on said missing data. Like with the update count, since the number of updates grows monotonically, eventually, all cookies will have enough updates so that there will be no missing entries. And therefore, the model trained on the data collected by the crawler cannot be applied to the data seen by the extension.

There is also another argument to be made. If we intend to ensure privacy, i.e., prevent hosts from collecting sensitive data, then the classification must occur immediately on first sight of the cookie. As such, we cannot wait for updates to arrive first to reach a conclusion, as this would give the host time to collect the sensitive information. Therefore, for the browser extension context, we use only a single per-update features for a cookie, which is its state when it was first created.

Third Party Cookies

Whether a cookie originates from a third-party cannot be determined when looking at cookies in isolation using the WebExtensions API. Only in special cases are cookies associated with a so-called `"first-party-domain"`, which is the domain the browser held in the address bar at the time the cookie was retrieved. One example is when the *first-party isolation* is activated in Firefox,

which unfortunately has undesirable side-effects and can potentially break website functionality [41].

A theoretical solution to approximate the third-party status would be to retrieve the address in the currently active tab when the cookie arrives. This is not completely sound however. In particular, consider the case where a user opens multiple tabs, or opens a tab in the background. This would cause first-party cookies to be wrongly identified as third-party.

Training Data Features

When porting the Python code to JavaScript, several small changes needed to be made for various steps in the process, which are too numerous to list here. To prevent such small differences from affecting the quality of the prediction, we needed to re-run the feature extraction on the training dataset using the JavaScript implementation, and then run the classifiers on this newly extracted feature matrix. This way, the resulting model cannot be conditioned on patterns that are absent for the JavaScript version of the cookie features. To redo the feature extraction for the training dataset, we implemented a simple *NodeJS* command line tool that included identical code as the browser extension. The resulting feature matrix was then ported back to the XGBoost Python implementation, using which a new model was trained.

5.3.2 CookieBlock Predictor

To implement the predictor, we utilize the XGBoost model dump. This is a JSON-formatted output of the forest of CART trees. The dump consists of four separate forests, each corresponding to one cookie purpose category. The nodes in a tree are represented as nested objects, containing the boolean decision function and feature index, which identifies the feature for which to perform a decision in each node. To reduce the filesize of the dump, we process it further to remove redundant information. All four forests together produce a model that is roughly 10 MB in size.

Predictions are made through a simple recursive function call that takes the cookie feature vector, as well as the model dump as input. Each recursive call traverses a node in the tree, until reaching a leaf containing the score for the input cookie. By summing the scores for each tree, we can compute the probability for the class. Finally, given the class probabilities, we reach a decision on the discrete label by taking the class with the highest probability.

Chapter 6

Evaluation

In this chapter, we present the performance of the crawler as well as the classifier approaches, concluding with a discussion of how the performance of the extension can be evaluated.

6.1 Crawler Performance and Analysis

In this section we present general results on the performance of the crawler, and analyze certain aspects of the collected training dataset, starting with an overview of the domains that were scraped for consent labels. Finally, in Section 6.1.4, we try to estimate the noise in the collected labels.

The Consent Management Platforms we selected for crawling are hereby *Cookiebot*, *OneTrust* and *Termly*. For more information on our selection process, we refer to Section 3.1.

6.1.1 Domain Sources

To train a classifier that can generalize well for any sample encountered in the wild, it is beneficial to have a large and unbiased training dataset. Because we are restricted to gathering training samples only from websites that make use of specific CMPs, this incurs some amount of bias into the dataset. We attempt to compensate this by selecting a varied set of domains to gather the cookies from.

Our primary source is Tranco [34], which we used to retrieve a list of the top 1 million domains by traffic worldwide,[1] as well as a list of 566 225 additional domains that are hosted in Europe, which are not contained in the top 1 million.[2] Tranco hereby is a domain ranking similar to Alexa, which

[1] Tranco list, dated 20 Nov. 2020, available at: https://tranco-list.eu/list/P5JJ
[2] Tranco Europe, dated 21 Nov. 2020, available at: https://tranco-list.eu/list/WNJ9

improves upon the shortcomings of other rankings, and moreover provides reproducibility, making it ideal for research purposes. We use European domains in addition to the top 1 million because we expect the top domains to be dominated by USA-based websites. With the GDPR being a EU regulation, we expect a greater degree of compliance, and thus a greater amount of consent notices among European websites.

Our smaller, secondary sources are Cookiepedia [45] with 44 942 domains, and BuiltWith [8] with 6956 domains. While these are not rankings like Tranco, they can be crawled for specific domains that have a high likelihood to contain one of the selected CMPs.

BuiltWith provides lead lists of web technologies for different geographic regions, such as rankings for Consent Management Platforms. For each CMP, they publicly list a small number of domains that they determined use the given technology. By scraping various region rankings on BuiltWith for Cookiebot, OneTrust, and Termly, we find a total of 6592 unique websites to add to the total list of domains.

Cookiepedia is a repository which lists purposes and descriptions for over 30 million cookies. In addition, it lists all domains where a cookie has previously been observed to exist. Both Cookiebot and OneTrust use a cookie to remember the user's consent choices, which is set as soon as the visitor arrives at the website. By querying Cookiepedia for these cookies and scraping the resulting list of domain, we receive a collection of websites that are highly likely to use OneTrust or Cookiebot CMPs. In total we scrape 41 932 domains with a OneTrust consent cookie, and 3010 domains with a Cookiebot consent cookie.

6.1.2 CMP Presence Crawler Results

In this subsection we present the results of the crawler described in Section 3.3.1, applied to the list of domains described in the previous section. To recapitulate, the presence crawler verifies whether a given domain is reachable, and if so, checks whether the landing page contains one of the supported CMPs. This reduces the number of domains that need to be targeted for the subsequent application of the consent label crawler. On a consumer-grade laptop with a 100 Mbit/s network connection and using a VPN located in the Netherlands, the crawl took approximately 23 hours to filter a list of roughly 1.6 million domains.

Table 6.1 lists the results. Note that there is some overlap in the domains that were gathered from Cookiepedia and Tranco. After removal of all duplicates, we receive a total of 9975 domains that potentially use Cookiebot, 20 380 domains that potentially use OneTrust, and only 665 domains that may use Termly. As the Termly domains are comparatively rare, the cookie

Table 6.1: Results of the presence crawler on the selected URL sources, with observed CMP market share among the reachable domains.

Domain List	Conn. Error	No CMP	Cookiebot	OneTrust	Termly
Tranco Worldwide	356 926	617 652	3929	5854	233
Tranco Europe	87 837	468 955	3849	1876	66
Cookiepedia	9392	15 745	1196	18 598	5
BuiltWith	590	2078	2194	1328	402
Total	454 745	1 104 430	11 168	27 656	706
Duplicates Removed	-	-	9975	20 380	665
		Market Share			
Tranco Worldwide	-	-	0.626%	0.933%	0.037%
Tranco Europe	-	-	0.811%	0.395%	0.014%

labels collected from this CMP will have a lesser impact on the resulting training. In total, we thus have 31 020 unique domains to scrape for the second stage of the crawl.

Observations

We observe a significant number of connection failures, especially in the Tranco top 1 million worldwide at 36.25%. Connection errors were generally either due to HTTP errors occurring, or a failure to establish a connection entirely. One possible explanation for this is bot-detection. Even though we use a fake user-agent string imitating a Chromium browser, it is still possible for websites to detect that we are connecting using the Python *requests* library. This could be solved by crawling the failing domains with actual browser instances, at the cost of computational time. Other possible reasons include include high latency, region blocks, or the domain set containing dead websites. These are observed consistently over multiple crawl attempts with the same list of domains.

As expected, only a small number of the crawled domains contain a CMP. Out of the reachable domains in the Tranco top 1 million, we observe a market share of 0.933% for OneTrust, and a share of 0.626% for Cookiebot. In the European domains, we see a share of 0.395% for OneTrust and a share of 0.811% for Cookiebot. As such, Cookiebot appears to be more common than OneTrust among the domains in Europe. Surprisingly, our expectation with the European domains is not fulfilled. In the Tranco top 1 million worldwide, roughly 1.60% of all domains use one of the selected consent notices, while in the European domains, we found only a share of 1.22%.

The market share results for the top 1 million match up remarkably well

with the observations made by Hils et al. [25] and the market shares found on BuiltWith [8]. This indicates that our crawler achieves comparable results to previous attempts to measure CMP presence on the Web.

6.1.3 Consent Crawler Results

In this subsection we present the results of the web crawler described in Section 3.3.2. In summary, this crawler uses OpenWPM [42] to retrieve cookies and their associated consent labels from websites that use specific CMPs.

Like the presence crawl, the consent label crawl was performed locally using a 100Mb/s DSL connection and a VPN located in the Netherlands. We crawled a total of 31 020 domains spread over 7 parallel browser instances. The browser instances were run on a 8-core Intel i7-8550U CPU with a clock rate of 1.80GHz per core. A full crawl for all filtered domains took roughly 84 hours to perform with this setup.

Domain Statistics

First, we will look at the success rate of the crawler on the target domains:

- For 26 403 out of 31 020 domains (85.1%), the crawl was able to successfully extract purpose labels and descriptions from the CMP.
- For 4371 out of 31 020 domains (14.1%), the crawl could successfully establish a connection, but could not extract any consent data. This is either because the CMP setup could not be parsed, or because there was no supported CMP on the website.
- For 246 out of 31 020 domains (0.79%), the crawl ended with a connection failure or HTTP error, preventing access to any data.

Additionally, for 2105 out of 31 020 domains (6.8%) the crawler eventually encountered a timeout while browsing the site. These cases are non-fatal, as some cookie data may still have been retrieved. In further detail, separating by CMP type, we get the following results:

- On 373 domains (1.2%), the crawler established a connection, but could not find any supported CMPs in the landing page.
- 9782 domains contained the Cookiebot CMP (31.5%). Out of these, 8272 allowed a successful data extraction.
  - The most common cause of failure with Cookiebot is due to an *"unrecognized referrer"* error. This occurred in 1159 out of 1510 failure cases. Such errors occur because the host made an error when setting up Cookiebot on their domain. We verified this by visiting the websites manually, and in all cases we examined, we found the same error in the browser log.

  - The second most common cause is that the crawler could not find the document that stores the consent information, which can occur if Cookiebot is only referenced in comments on the site, and not actually active. This happened in 264 cases.

- OneTrust is found on 20 032 domains in total (64.58%). Out of these, 17 823 lead to a successful data extraction.
  - The most common cause of error with OneTrust occurred because the cookie data was only available in an unsupported language, which makes up 1488 out of 2209 failure cases.
  - The second cause is because the crawler could not identify how to extract the cookie information from the OneTrust CMP, which occurred in 538 cases. This can either be an indication that there exists a OneTrust implementation which our crawler does not handle, or that the CMP plugin is not installed properly.
  - The third cause occurred due to connection failures when trying to access the remote OneTrust domain. In 111 cases, this lead to a HTTP error. This is an indication that OneTrust does not recognize the domain, and the CMP is hence not working correctly on the crawled website.
- Termly is found on 587 domains in total (1.9%). Only from 308 domains could the CMP data be successfully extracted, further reducing the number of cookie labels retrieved for this CMP.

In summary, our consent crawler reports a success rate of 85.12%. We find that the majority of the failed crawls originated from host errors that out of our control. A minority could be traced to errors in the crawler implementation, which have since been resolved.

Cookie and CMP Data Statistics

In this subsection we present the overall statistics on our collected consent label data, as well as the actual cookies that were retrieved from the websites. During the crawl, we collected a grand total of 622 005 cookies. The average number of cookie updates was 3.63 updates. Separated by CMP we get:

- *Cookiebot*: 174 302 cookies (28.02%). Average: 17.82 cookies/site
- *OneTrust*: 437 846 cookies (70.39%). Average: 21.86 cookies/site
- *Termly*: 9857 cookies (1.58%) Average: 16.79 cookies/site

In terms of data scraped from the content of CMP documents, we collected a total of 1 715 700 consent label entries, which consist of a total of 778 909 unique cookie identifiers, of which at least one third is third-party cookies.

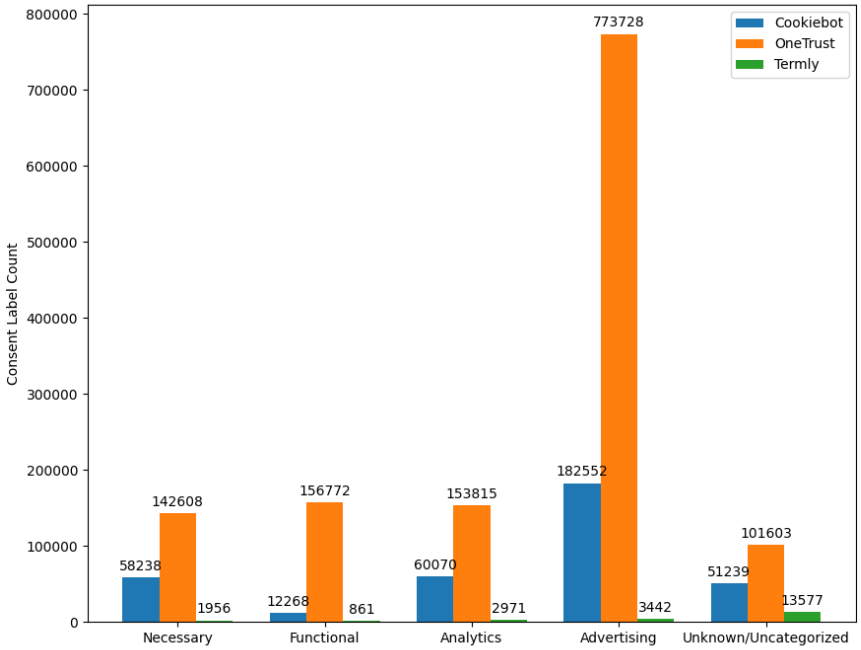

Figure 6.1: Number of collected consent labels, separated by category and origin.

As such, a large number of cookies that are declared in consent notices are also from third-parties. We find the following label counts:

- *Strictly Necessary*: 202 802 (11.82%)
- *Functionality*: 169 901 (9.90%)
- *Performance/Analytics*: 216 856 (12.64%)
- *Tracking/Advertising*: 959 722 (55.94%)
- *Unclassified/Unknown*: 166 419 (9.70%)

Figure 6.1 gives a more detailed breakdown of these results, with each category additionally separated by origin CMP. We observe that the overwhelming majority of unique cookie declarations belong to the advertising class. Among those, the overwhelming majority stems from the OneTrust CMP, but also Cookiebot declares a majority of its cookies with this purpose. We also observe that "Functionality" is the least commonly declared category, and that it is particularly uncommon in the Cookiebot CMP. For cookies with an "unknown" categorization, most of the entries from OneTrust stem from unrecognized category names, for example due to an unsupported language, while for Cookiebot and Termly, this stems from unclassified cookies. Termly declarations are not only very rare but also unspecific. In fact, more

than half of all cookie declarations we found for Termly actually do not assign any purpose, but instead leave it as "unclassified".

Cookie/Label Count Discrepancy

There exists a large discrepancy in the number of actual cookies found versus the number of declared cookie labels. In particular, we observe almost three times as many cookie declarations than actual cookies. Yet at the same time, as we will see in Sections 6.1.3 and 7.5, only a fraction of the cookie declarations actually match a real cookie.

Splitting the results up by CMP type, we find that 16.45% of the declarations that do not match an actual cookie originate from Cookiebot, 82.10% originate from OneTrust, and 1.45% originate from Termly. As such, the majority of unobserved cookies stem from OneTrust. We observe an average declarations per site of 44.05 for Cookiebot, 74.54 for OneTrust, and 58.78 for Termly.

We can think of a number of explanations for this phenomenon. The first is that the number of declarations is over-inflated compared to the cookies that are actually present on the website. It is possible that the CMP simply declares all cookies that are offered by third-party vendors, rather than the ones that are used. Another possibility is that the consent notice not only includes cookies, but also other types of tracking technologies. This is at least the case for Cookiebot, which also explicitly declares tracking pixels.

The second explanation is that our crawl does not interact with the website in a complex manner. For example, no website settings are changed, no special buttons or widgets are interacted with and the crawler never registers or logs into an account on any of the websites visited. This is likely to decrease the number of observed cookies, specifically those that are declared as "functional" or "necessary". This is also reflected by the comparatively low number of functional cookies we observe in the training samples, as described in the following subsection.

Training Sample Statistics

After matching the consent labels with the actual cookies, as explained in Section 3.4, we receive the dataset of training samples. Similarly to the previous, this subsection presents statistics on this data.

In total, we have 309 472 samples in the final training dataset. There is an average number of 4.51 updates per training cookie, and the most updates we have seen for a single cookie is 2853. It was not possible to pair roughly half of the collected cookies with a matching declaration, an issue we will investigate further in Chapter 7. A detailed breakdown of the training data counts, separated by origin and purpose, is given in Figure 6.2.

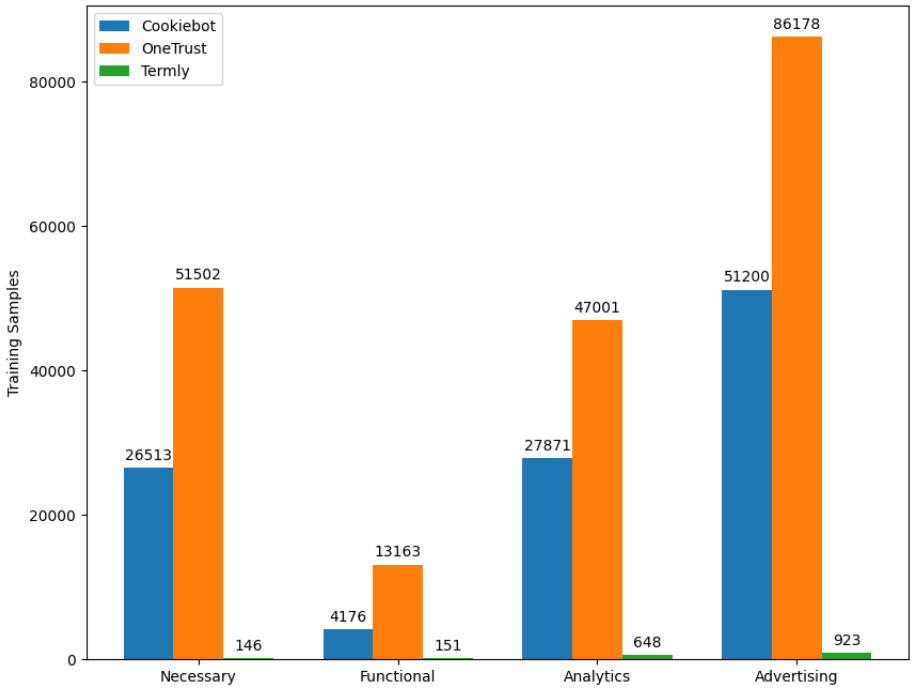

Figure 6.2: Number of training samples, separated by category and origin.

For the classifier, class imbalance can be an important factor to consider during training. The number of samples for each purpose category are:

- *Strictly Necessary:* 78 161 (25.26%)
- *Functionality:* 17 490 (5.65%)
- *Performance/Analytics:* 75 520 (24.40%)
- *Tracking/Advertising:* 138 301 (44.69%)

As we can see, the advertising category still has a majority, although it is less dominant than for the consent labels. What is particularly concerning is the low number of observations that we have for functionality cookies. At just 5.65%, we lack a significant amount of training data for this category. This is not easy to compensate, and will likely make the classifier particularly inaccurate for "Functionality" type cookies.

After extraction of the training data, the number of Termly cookies decreases even further to almost insignificant levels. With such a low number of useable samples, it may not be worth crawling additional CMPs with a similar market share as Termly.

Another observation we can make is on the mean and standard deviation

of cookie updates per category label. A cookie update hereby occurs any time a HTTP request or JavaScript call tries to create a cookie that is already present in the browser, regardless of whether the contents are altered:

- *Strictly Necessary*: 5.24 ± 12.30 updates per cookie
- *Functionality*: 5.87 ± 21.55 updates per cookie
- *Analytical*: 5.37 ± 6.95 updates per cookie
- *Tracking/Advertising*: 3.13 ± 5.15 updates per cookie

Cookies that serve functional purposes appear to be updated more frequently on average than tracking cookies, and moreover can have a highly variable number of updates. Tracking cookies see less updates than the other categories, and also hold the lowest standard deviation. These measurements could help a classifier distinguish between the classes.

6.1.4 Lower Bound Estimate for Label Noise

Consent category labels collected from Consent Management Platforms can be unreliable, as individual hosts may misclassify cookies, either intentionally or by mistake. At the same time, it can provide us with a diverse set of cookies whose labels are retrieved directly from the hosts that created them. This may include rarely observed cookies for which repositories such as *Cookiepedia* do not provide any information.

In this section, we discuss how we can estimate a lower bound for the degree of noise of the labels in our dataset. To do so, we attempt to determine the majority class for third-party cookies. If this majority class is correct, then the outlier labels count towards the total number of errors. If instead the majority class is wrong, then the number of wrong labels can only be strictly larger.

Overall, 120 726 of the 309 472 cookie samples occur more than once, and are therefore third-party cookies. Out of these samples, 8744 have labels that deviate from their respective majority class. This gives us a lower bound of 7.24% of labels that are noise among the third party cookies, and a lower bound of 2.83% over all training samples. Overall, there appears to be a strong agreement across the websites we crawled, which may be a result of CMPs suggesting default cookie labels to their customers [13, 50]. This gives us confidence that our ground truth is not strongly perturbed by noise.

Note that this lower bound only measures the noise for third-party cookies. First-party cookies may also be mislabelled. However, this is harder to measure, as they are often unique and must be evaluated on a case-by-case basis. We expect the number of wrong labels to be lower for first-party cookies than for third-party ones however, as with these cookies being rarely used for tracking, there is less of an incentive to misguide the visitor.

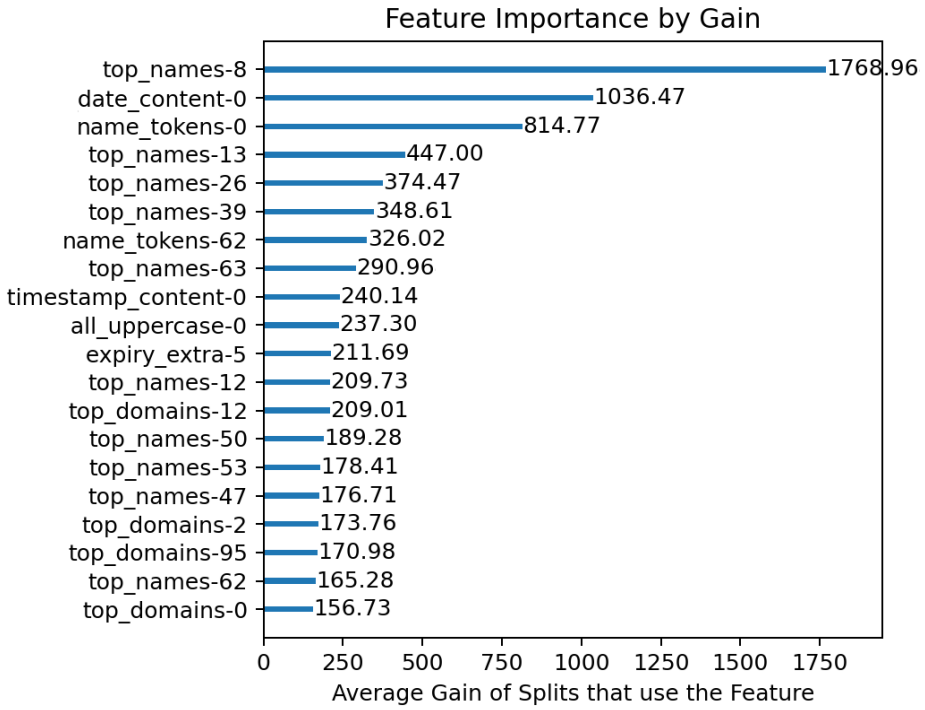

Figure 6.3: Importance of the top 20 features by gain. This measures how high the increase in "decision power" is with the introduction of one of the listed features.

6.2 Feature Evaluation

In this section we will use the feature importance, as reported by XGBoost, to analyze which features are most useful in representing the domain of cookies. We utilize two different importance measurements, namely the *gain* and *weights* metrics.

6.2.1 Feature Importance by Gain

The *gain* represents the increase in expressiveness and decision power in the model when splitting on a feature. In other words, this metric represents how much a specific feature increased the model's ability to distinguish between the four cookie purpose classes.

In Figure 6.3 we show the importance of the top 20 features when using the *gain* metric. The y-axis displays the names of the features with associated vector index, while the x-axis is the gain of the feature. The higher the gain, the better the feature allows the model to distinguish between classes.

We observe that the name and domain one-hot vectors are among the most

important features of the training dataset. These are categorical features that determine whether the name or domain of the cookie match one of the most commonly found names and domains, as determined by a separately constructed ranking.

In particular, the features "`top-names-8`" and "`top-names-13`" represent the cookie names "`__cfduid`" and "`__gads`", respectively. In addition, the "`name-tokens-0`" feature represents the "`gat`" Google Analytics identifier. These features make sense to be useful to the classifier, as unique identifiers and Google Analytics tokens are indicative of tracking and analytics cookies respectively. A possible concern with these features is that they may be too specific, in the sense that they may cause the classifier to overfit to the specific origins and names commonly found in the training set. To prevent this, we sourced the top name and top domain features from a separate, randomly selected set of 10 000 domains that were crawled to gain random cookie data.

In terms of more generic features, we find that dates, timestamps and the expiration time appear to be important. Whether a cookie consists of all uppercase characters also appears to play a significant role in being able to distinguish between purpose classes.

6.2.2 Feature Importance by Weight

The *weight* metric is the total number of nodes in all forests that use the specified feature to perform a decision. It indicates which features the classifier training relied most on.

In Figure 6.4, we list the importance in the top 20 features using the *weight*. Unlike with the gain, here we mostly see generic features being used, in particular the entropy, the expiration time, and similarity between cookie contents. This indicates that these features still play an important role in the training to reach more nuanced conclusions.

6.2.3 Auxiliary Feature Analysis

In the following we verify whether certain flags and contents of a cookie can actually be observed to be altered when the cookie is updated. If not, then features that check for such changes would be of little use. Over the 309 472 unique training samples, we have found:

- 2402 instances (0.78%) where the expiry was changed by more than 24 hours.
- 795 instances (0.26%) where a persistent cookie turned into a session cookie, or vice-versa.
- 327 cases (0.11%) of the secure flag being toggled in an update.

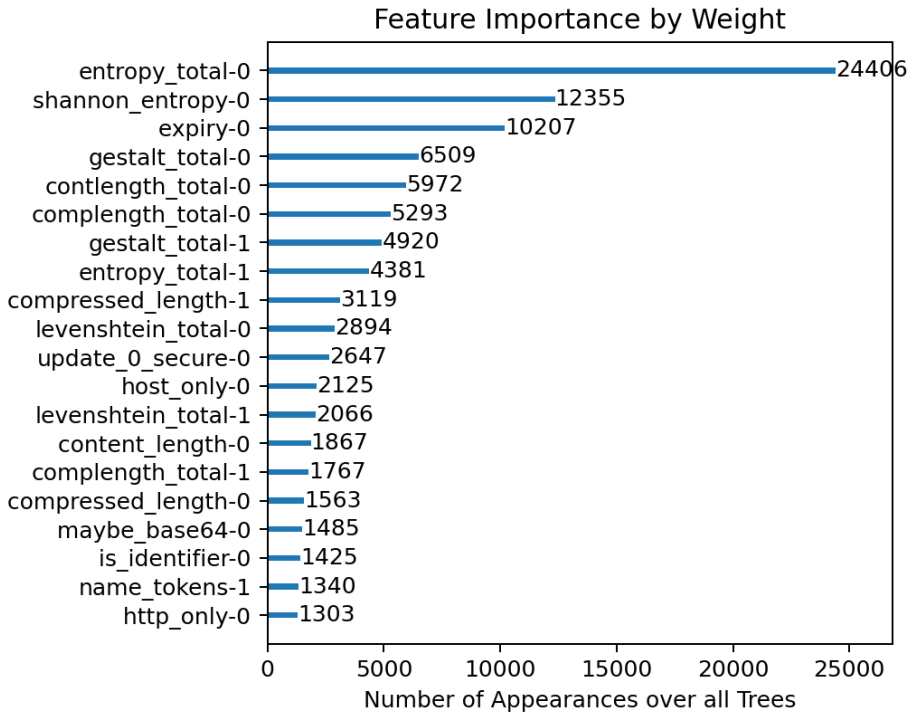

Figure 6.4: Importance of the top 20 features by weight. This measures the number of times a feature was used in the forest of trees overall.

- 201 cases (0.06%) of the SameSite flag changing in an update.
- 146 cases (0.05%) of the HTTP-Only flag changing in an update.

This proves that the select flags and the expiry can change in a cookie update. We also observe that the content of a cookie changed in 39 798 out of 309 472 cases (12.86%). While rare, such features could still provide a significant gain in helping the classifier distinguish between cookie purposes.

6.3 Classifier Evaluation

In this section we present the classifier performance in the offline setting, using the collected dataset for both training and validation. We will first recall some terminology that is useful for the following evaluation. We discuss how we obtain a baseline comparison, and then present our results.

6.3.1 Terminology

In this subsection we briefly recall the terms of *confusion matrix*, *precision* and *recall* to support the following evaluation.

In a multi-class classification problem, the *confusion matrix* C visualizes the accurate and inaccurate predictions by each ground truth class. We designate the ith row of the matrix C as the ground truth, and the jth column as the predicted label. Then, the entry C_{ij} determines how many instances of ground truth i were assigned class j. Note that the diagonal in the confusion matrix represents the accurate predictions for each class.

The *precision* of a class k represents how many instances that were assigned the label k actually matched the ground truth. For instance, if we assigned the label "Strictly Necessary" to 100 cookies, and only 30 of these are actually used for this purpose, this gives us a precision of 30% for that class.

The *recall* of a class k represents how many of the total instances that had the ground truth k were assigned the correct label. For example, if there exists a total of 100 functional cookies in the dataset, and we assign the "functionality" category to 60 of these, then we reach a recall of 60%.

6.3.2 Cookiepedia Baseline

To the best of our knowledge, we are the first to apply machine learning techniques to the task of categorizing cookies into purposes. As such, we can only compare our approach to manual classification efforts, performed by human operators.

What we consider the state-of-the-art in this regard is the publicly accessible database *Cookiepedia* [45], which reportedly records over 30 million cookies. For each cookie, the repository lists the name, host, path, lifespan, and whether the cookie is flagged as secure or HTTP-only. In addition, a team of human operators determines the purpose of individual cookies, manually classifying them and attaching a description. Since this is a gradual process, not every cookie in the repository has a purpose label assigned to it.

To utilize Cookiepedia as a quasi-predictor, we send a query for each cookie name in our dataset and obtain the corresponding label from the repository. This label is then compared to our ground truth, which are the labels we collected from the Consent Management Platforms.

As is normally the practice in cross-validation, we evaluate Cookiepedia's prediction accuracy, precision and recall by splitting the dataset of 309 472 training cookies into five equally-sized folds, which serve as the test sets. Using the extracted label and the ground truth, we construct a confusion matrix and compute the precision and recall for each fold. Finally, we compute the mean and standard deviation of those metrics over all five folds.

It may occur that Cookiepedia does not know a class for the given cookie name, either because the cookie is not stored in the database, or because no human operator has assigned a label to it yet. We record the number of

Table 6.2: Cookiepedia precision and recall per category, evaluated using a 5-fold split. Total number of cookies with available category: 77.82%

Cookiepedia – Average Accuracy: 85.46% ± 0.14%				
	Necessary	Functional	Analytics	Advertising
Mean Precision	93.18%	36.86%	81.33%	94.40%
Std.Dev. Precision	±0.14%	±0.90%	±0.30%	±0.12%
Mean Recall	87.72%	73.26%	91.57%	81.10%
Std.Dev. Recall	±0.20%	±0.96%	±0.12%	±0.15%

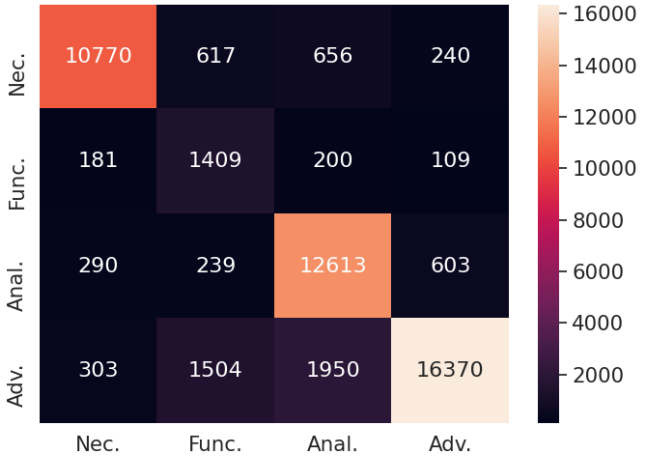

Figure 6.5: Confusion matrix heatmap for a single fold of the Cookiepedia baseline. The rows represent the ground truth, while the columns represent the labels from Cookiepedia.

missing labels, and report the overall cookie coverage. To keep the comparison with the classifier approaches fair, we do not factor the missing cookie labels into the precision/recall computation.

Baseline Performance

Table 6.2 presents the results of the Cookiepedia baseline. The average accuracy that Cookiepedia achieves over all classes is 85.46%, with a standard deviation of 0.14%. It shows a particularly high precision for "Necessary" and "Advertising" type cookies, and a high recall for "Necessary" and "Analytics". Additionally, Cookiepedia stores categories labels for a total of 77.82% of the cookies in our dataset. We also see that the precision and recall per class deviates little depending on the fold.

Particularly interesting here is the low precision for "Functionality" type

cookies. To gain further insight on why the precision is so low, we analyze the confusion matrix in Figure 6.5. Here, we see that a large number of cookies with ground truth "Necessary" and "Advertising" were assigned to "Functionality" instead.

Ultimately, the reason for why precision for functional cookies is low is due to the class imbalance. There exist significantly more cookies for all other categories, meaning that if a fraction of these are misclassified as functional, the effect on the precision of this category will be significant.

6.3.3 Classifier Performance

We now present the performance of each classifier we applied. For each model, we performed a random search to find a good combination of hyper-parameters, and we applied 5-fold cross-validation to produce the average performance metrics.

Each model was trained for a maximum of 2000 boost rounds with early stopping after 30 rounds of observing no increase in the validation score. Each boost round constructs a new tree, meaning that the number of rounds also represent the size of each forest. Validation was performed using the average error and the multiclass cross-entropy as metrics, with the latter being used for early-stopping. We set the maximum tree depth to 12 levels, and used a learning rate of 0.3.

For XGBoost, the training took on average 6.5 minutes per fold, and per fold, the number of actual boost rounds was in a range of 167 to 322 rounds, with an average of 244. The average accuracy is 86.90% with a deviation of 0.12%.

With LightGBM, training was much faster than XGBoost, and only took 49 seconds on average per fold. We trained trees in a range of 162 to 209 trees per forest, and achieved an average accuracy of 86.16% with a standard deviation of 0.10%.

Finally, CatBoost took the longest to train by far, with 26.5 minutes on average. The long training time is a result of training for the entire 2000 boost rounds in each fold, as the validation score continuously improved by small margins. The average accuracy for CatBoost was 87.21% with a standard deviation of 0.19%.

An overview of the precision and recall for all evaluated approaches, including the Cookiepedia baseline, is presented in Table 6.3.

Comparison and Summary

First, we see that each of our classifiers outperforms Cookiepedia on the collected dataset in terms of average accuracy. CatBoost achieves the best

Table 6.3: All precision and recall for XGBoost, LightGBM, and CatBoost using 5-fold split. The Cookiepedia baseline results are also listed to simplify direct comparisons.

	Necessary	Functional	Analytics	Advertising
Cookiepedia – Average Accuracy: 85.46% ± 0.14%				
Mean Precision	93.18%	36.86%	81.33%	94.40%
Std.Dev. Precision	±0.14%	±0.90%	±0.30%	±0.12%
Mean Recall	87.72%	73.26%	91.57%	81.10%
Std.Dev. Recall	±0.20%	±0.96%	±0.12%	±0.15%
XGBoost – Average Accuracy: 86.90% ± 0.12%				
Mean Precision	86.46%	48.54%	87.19%	94.97%
Std.Dev. Precision	±0.50%	±1.27%	±0.24%	±0.28%
Mean Recall	80.52%	76.96%	87.66%	90.25%
Std.Dev. Recall	±0.19%	±1.48%	±0.21%	±0.30%
LightGBM – Average Accuracy: 86.16% ± 0.10%				
Mean Precision	86.72%	44.66%	87.38%	95.15%
Std.Dev. Precision	±0.48%	±0.79%	±0.19%	±0.16%
Mean Recall	78.77%	79.62%	86.78%	89.57%
Std.Dev. Recall	±0.20%	±0.48%	±0.39%	±0.14%
CatBoost – Average Accuracy: 87.21% ± 0.19%				
Mean Precision	85.81%	54.20%	86.41%	93.61%
Std.Dev. Precision	±0.55%	±0.79%	±0.44%	±0.14%
Mean Recall	81.67%	68.65%	88.08%	91.25%
Std.Dev. Recall	±0.62%	±1.02%	±0.39%	±0.17%

performance at 87.21%, but is also the classifier that took the longest to train, with the most trees per forest. LightGBM achieves a worse performance than XGBoost and CatBoost, but can be trained significantly quicker than either of the two.

Compared to all of our classifiers, Cookiepedia has an advantage in both precision and recall with the class of "Strictly Necessary" cookies. A lower precision for this purpose can indicate less privacy, as this could mean that more tracking/advertising cookies are wrongly interpreted as being strictly necessary. Furthermore, a lower recall can indicate that more websites break, as strictly necessary cookies are wrongly classified and possibly removed by the extension. Cookiepedia achieves a better precision at 93.18%, and a better recall at 87.72%, while our classifiers only achieve a maximum of 86.72% and 81.67% respectively.

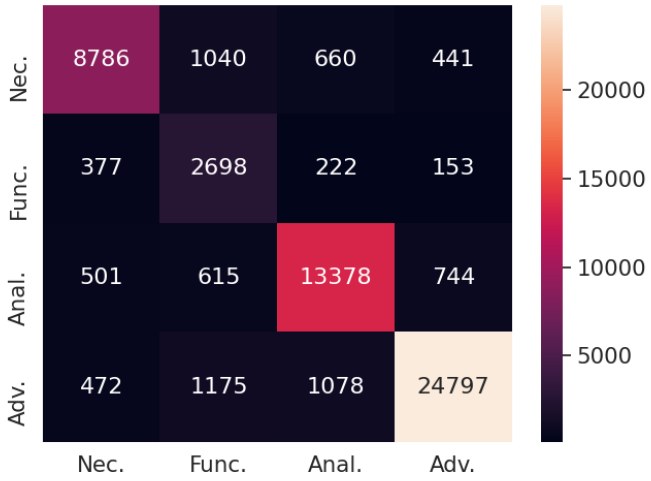

Figure 6.6: Confusion matrix heatmap for a single fold of the XGBoost evaluation. The rows represent the ground truth, while the columns represent the predicted labels.

By viewing the confusion matrix for XGBoost in Figure 6.6, we see that this issue is less severe than it first appears. Namely, the main difference compared to Cookiepedia is that our classifiers assign more necessary cookies to the "Functionality" class. 9.52% of the "Necessary" labels were assigned to "Functional" cookies, 6.04% were assigned to "Analytics", and 4.04% are assigned to "Advertising". This lies within the expected results, as functional cookies are often difficult to distinguish from strictly necessary ones.

The applied classifiers are able to achieve increased precision and recall for "Functionality" type cookies compared to the Cookiepedia baseline. Moreover, we also achieve a higher precision for "Analytics" type cookies, as well as a higher recall rate for the "Advertising" class. The remaining metrics are comparable, and the standard deviation is consistently low across all folds.

LightGBM is able to train the model much quicker than either of the other approaches, at the cost of overall accuracy. This could be useful in case the number of features is increased, for example by further increasing the size of the "*Top Names*" and "*Top Domains*" categorical feature vectors. It would also be useful if more cookies are gathered for classifier training, for instance with an approach that trains on the entire Cookiepedia dataset of cookies.

CatBoost achieves the best overall performance. However, due to the high increase in training time and the size of the resulting forests, this small increase does not seem worth the cost. The number of trees in the forest is a factor that is important to keep in mind for the purpose of a browser extension, as it directly affects the filesize of the resulting model.

In our case, the resulting CatBoost model size was 89 MB, while the XGBoost model had a filesize of roughly 10 MB. This makes the latter much more suitable for integration into the extension.

Finally, one major advantage of our approach over Cookiepedia is that it is fully automated, and can be applied to any cookie found in the wild. Cookiepedia only covers 77.82% of all cookies in our dataset, while we achieve full coverage. No further human input is required to predict a purpose for new cookies, thus massively speeding up the process. The downside compared to Cookiepedia however is that we cannot generate a detailed, human-readable description of each cookie, which may also be required to allow the visitor to make informed decisions.

In conclusion, each of the applied tree-ensemble models can outperform the state-of-the-art in cookie classification, based on the dataset we collected. The best approach, CatBoost, achieves an increase of 1.75% in overall accuracy over Cookiepedia. However, due to the size of the forest produced by CatBoost, we instead opted to use the smaller XGBoost model for integration into the browser extension.

6.4 Extension Evaluation

In Chapter 5 we described the implementation of CookieBlock, which is intended to ensure that a user's cookie consent preferences are applied to each website that is visited.

In this regard, there are three criteria that can be evaluated:

- *Effectiveness at ensuring user privacy*: While our applied classifiers all outperform the baseline on our collected dataset, this does not guarantee that the extension is effective at ensuring user privacy. In order to verify this, it is required to evaluate the predictor on known tracking cookies that were gathered from a random sample of websites, and to then verify that the predicted label is accurate.
- *Preservation of website functionality*: It should also be verified that the extension does not inadvertently break essential functionality of websites by removing cookies that are strictly necessary. For this, we need to perform extensive testing on a variety of websites using a multitude of different configurations.
- *Usability*: To ensure that the extension is easy to use, and moreover does not negatively affect the browsing experience, we need to perform a user study with a small group of dedicated testers, and gather feedback on these aspects. One particular concern for instance is the speed of the feature extraction and the label prediction, and whether the extension can slow down the browser in special cases.

Table 6.4: Precision and Accuracy metrics of the predictor implemented in JavaScript, using the features extracted through NodeJS, see Sections 5.3.1 and 5.3.2. Uses the XGBoost model.

	Necessary	Functional	Analytics	Advertising
Precision	88.18%	43.94%	86.31%	94.32%
Recall	76.96%	74.29%	87.96%	89.84%

Unfortunately, we did not have time to perform these evaluations in the scope of this master thesis. However, they will likely be explored in future work involving CookieBlock.

In the remainder of this section, we will present the performance of the JavaScript predictor combined with the reduced feature set, as described in Sections 5.3.1 and 5.3.2.

6.4.1 CookieBlock Predictor Accuracy

In Chapter 5, we described how we altered the feature extraction for the training samples, and how we reimplemented the tree predictor for the purpose of classifying cookies within the extension. For this, the code had to be translated from Python to JavaScript, and a number of features needed to be altered to accommodate for the new setting. This included the reduction of the per-update features down to a single update, as well as the removal of the third-party indicator feature.

In this subsection, we analyse how this change affects the performance of the XGBoost model. Table 6.4 presents the precision and recall results for the CookieBlock predictor, evaluated on a single validation set of cookies.

The overall accuracy of the XGBoost predictor dropped from 86.90% to 85.92% after the changes were applied, which still outperforms Cookiepedia. The overall performance was reduced slightly in precision and recall, but the results still lie within the standard deviation. Therefore, the changes we applied to the feature extraction and the predictor for the browser extension did not reduce the performance to a degree that would make the predictions significantly worse than in the offline setting.

To summarize, we can confidently say that CookieBlock outperforms the state-of-the-art in cookie classification accuracy, at least on the cookie labels we scraped from different CMPs. The offline classifiers written in Python do give a slightly better performance, which can be useful for predicting labels for large datasets of cookies, and analysing the results.

Chapter 7

Automatic Violation Detection

Using knowledge of specific articles of the GDPR and related court rulings, we determine six novel methods to help identify inconsistencies inside consent notice designs, and to detect potential violations of current legislation which expands on existing work in the area. This is intended to help supervisory authorities in ensuring compliance, and to provide ways in which researchers can determine the extent of non-compliant behavior.

These methods require the extraction of consent labels from the consent notice found on a website. In our case, this is limited to websites that host the *Cookiebot*, *OneTrust* or *Termly* CMPs, but this can in principle be expanded. We apply these methods to our own collected dataset, which consists of a total of 1 715 700 consent label declarations and 552 454 cookies that were found on 26 403 websites with supported CMPs.

Our methods include the detection of:

1. Wrongly assigned labels for cookies with a known purpose. For *Google Analytics* cookies, we observed this on 15.46% of all domains.
2. Outlier labels, which deviate from the majority opinion of other websites that held the same cookie. Observed on 27.75% of all domains.
3. Incorrect retention period, i.e., cookie expiry. This involves cookie expiration dates that are at least 1.5 times greater than the declared expiry time. Observed on 5.30% of all domains.
4. Unclassified cookies. Cookies that are declared, but which have not been assigned a category. Observed on 19.21% of all domains.
5. Undeclared cookies. Cookies that are found on a website, but which do not appear in the consent notice. Observed on 86.08% of all domains.
6. Declarations with contradictory labels. Seen on 3.51% of all domains.

At least one of the above listed pieces of evidence can be observed on 91.87% of all crawled domains. If we exclude the undeclared cookies, we observe the remaining evidence on 49.30% of all websites.

Note that the evidence collected in this chapter may not necessarily point towards malicious intent. In many cases, particularly for undeclared cookies, the host may simply not be aware of the cookies that are present on the website, or their correct purpose. Still, non-compliance by mistake is ultimately still non-compliance, and what is presented here shows how strictly the rules of the GDPR are being followed on the web.

The remainder of this chapter will proceed as follows. In each of the following six sections, we will discuss one of the aforementioned methods in more detail, and present additional results. To conclude, we describe two as of yet unimplemented approaches to identify potential violations, which could reaffirm observations made in previous work.

7.1 Method 1: Wrong Label for Known Cookie

Recital 32 of the GDPR [22] states that consent must be specific and informed. Furthermore, in a decision by the EU Court of Justice on the Planet49 case [30], it was made clear that a website host cannot declare "Google Analytics" cookies as being strictly necessary for the operation of the site.

Based on these decisions, if a website wrongly declares a cookie for which the purpose is well-known, such practice could be considered deceitful, and thus may be in violation of the requirement of specific and informed consent. Furthermore, the ePrivacy directive states that strictly necessary cookies do not require consent [21]. Therefore, if a known cookie is falsely labelled as "*strictly necessary*", then in most consent notice implementations, the user will be forced to accept that cookie.

"Google Analytics" (*GA*) cookies are some of the most common first-party cookies found on the web. As the name implies, they serve the purpose of Web-Analytics. In our dataset alone, at least 75 454 cookie declarations (4.40%) belong to GA. These are usually first-party cookies that have names such as `"_ga"`, `"_gat"`, `"_gid"`, among others. By querying our database for all websites that mislabel these cookies, we can gather evidence for potentially deceptive consent notices.

We analyze the number of mislabeled GA cookies in our dataset to demonstrate this approach. We observe that 15.46% of all domains in our dataset fail to assign at least one GA cookie to the proper purpose. Out of 75 454 GA cookie declarations, 12 849 (17.02%) were assigned the wrong category by the host of the website. Moreover, at least 2172 (2.88%) GA cookies were

declared as "Strictly Necessary", meaning that they were set in the browser before any consent could be given.

This approach is of course not just restricted to GA cookies. In principle, it could be repeated for any well-known cookie of a particular purpose. We leave this open as potential future work to be pursued.

7.2 Method 2: Identifying Outlier Labels

The second method of gathering evidence for potential GDPR violations is similar to the first. Again we try to determine potential misclassifications of cookies, but this time also for those that do not have a definitively known purpose. Namely, we approximate the ground truth by using the majority label of that cookie in the dataset, similarly to the lower-bounds noise analysis described in Section 6.1.4. Because the cookie needs to be seen across multiple domains, this method is only applicable to third-party cookies.

First, for each unique cookie identifier, we count the number of occurrences for each label, and then select the majority. To prevent a majority from being too narrow and undecisive, we restrict the analysis to cookies that occur at least 10 times across different domains, and where the most common label has at least a two thirds majority. Then, we simply identify all domains that assign a label to this cookie that does not match the majority opinion.

In total, 27.75% of the analysed domains contained at least one outlier cookie label. A total of 26 371 cookie declarations deviate from a majority opinion. In 5046 cases, a cookie was declared as "Strictly Necessary" when the majority label for that cookie was "Advertising" or "Analytics". These are particularly severe cases of mislabeling, as consent notices will normally not request consent for strictly necessary cookies.

When used for determining potential violations, the results produced by this method should be further analysed by a human operator, namely to determine whether the majority opinion for that cookie is actually correct. In the case where the majority is incorrect, this may be indicative of a CMP suggesting a wrong default label to its customers. The Cookiebot and OneTrust CMPs in particular use label repositories to suggest default categories for common cookie identifiers [13, 50].

7.3 Method 3: Incorrect Retention Period

Article 13 of the GDPR [22] and the Article 29 Working Party (29WP) [2], an EU body which advises on the interpretation of the EU cookie directive, define the necessary information that needs to be declared in relation to cookies. This includes the purposes for processing personal data (Article 13 1(c)),

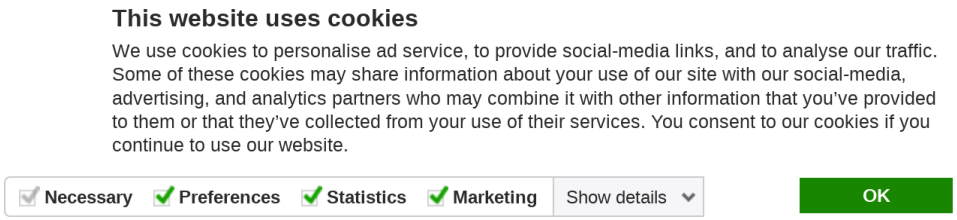

Figure 7.1: The consent choices offered by the *Cookiebot* plugin. Note that unclassified cookies, while listed in the drop-down menu, cannot be enabled or disabled here.

the third parties with whom the data is shared (Article 4(9)), as well as the *storage period* of the cookie (Article 13(2)(a)). In particular, France's CNIL [11] proposes guidelines on the maximum lifetime of cookies, corresponding to their purpose [47].

A failure to accurately declare the expiration time for cookies may thus also violate the rules set up by the GDPR. To do so, we record each cookie for which the actual expiry is more than 1.5 times the declared period, as well as the website that held the corresponding cookie. Furthermore, any cookie that is declared as a session cookie, but is actually persistent will also be flagged, and vice-versa.

We found cookie expiration date inconsistencies in 5.30% of the total domains, indicating that this requirement appears to be relatively strictly adhered to.

Note that for this analysis, we specifically excluded the cookie with the name "*CookieConsent*" from these results, as it introduced a strong bias. This cookie stores the consent preferences of the visitor for Cookiebot, and was thus found on all of the domains that used the Cookiebot plugin. On all instances we found, the cookie exhibited an expiration date that deviated from the declared duration by several years. However, as soon as the user interacted with the consent notice, the cookie would correct itself. Therefore we ignored this in the analysis.

7.4 Method 4: Unclassified Cookies

Article 7 as well as Recital 32 of the GDPR state that consent must be specific and informed in order to be considered valid. In effect, this means that a consent notice must declare and describe the purpose for which personal data is collected, and why cookies are stored on the user's browser.

This is not always the case however. Some websites, despite using a CMP to achieve compliance with the GDPR, nevertheless neglect to assign some of their cookies or other browser tracking technologies to any purpose, leaving them without a description. This is more likely to be a result of neglect

rather than malice. In one support article, the maintainers of the Cookiebot CMP explain that for cookies for which a purpose cannot be determined, the purpose will either have to be requested from the third party that sets them, or the host will need to be classify them manually [50]. This suggests that unclassified cookies are a result of laziness more than an intentional attempt at deceiving the user.

Unclassified cookies were found on 19.21% of the examined domains. Moreover, 64 833 (3.78%) of all declarations were unclassified. The majority of these originate from Cookiebot with 51 239 (79.03%) entries, and Termly with 13 097 (20.20%) entries. It appears that websites that use the OneTrust CMP are more diligent with assigning cookies to purpose categories and describing their purpose.

Note that Cookiebot does not allow accepting or rejecting consent for unclassified cookies, see Figure 7.1. The resulting behavior is intransparent and inconsistent – on some websites, the declared unclassified cookies can never be observed, while on others, they behave just as if they were part of the other categories, and as if they had been consented towards.

7.5 Method 5: Undeclared Cookies

In this subsection, we denote as *undeclared cookies* those which can be observed being set inside the browser, but which at the same time are not listed in a website's consent notice. Similarly to unclassified cookies, undeclared cookies are indicative of the user not being fully informed on the data that is being stored inside the browser. As such, they may indicate that a website is in violation of informed consent, especially in cases where some, but not all cookies are listed.

We observe undeclared cookies on a large majority of 86.08% of all analyzed websites. Out of the 552 454 cookies that originate from websites with one of our supported CMPs, 233 179 (42.21%) could not be matched with a corresponding declaration. We can think of three likely reasons:

1. The first is that some of these cases stem from the declared name or domain not exactly matching the identifiers found in the actual cookies themselves. For example, by only matching the declaration and cookie on the name, and ignoring the domain in the process, this reduces the number of domains with undeclared cookies to 80.79%, and the number of undeclared cookies to 188 845 (34.18%).
2. The second reason is that hosts are not fully aware of the cookies that are hosted on their websites, and that tools that are designed to detect such cookies do not achieve full coverage.

3. The third reason may be that the hosts simply do not care, and do not list all the cookies as there is a very low chance of facing any fines for doing so.

In either case, we presume that many of these missing declarations originate not from malice, but from neglect, and a lack of accuracy in the declaration.

7.6 Method 6: Contradictory Labels

An oddity we encountered during the extraction of the training data involved cookies that were assigned two different labels originating from the same domain. In some cases, a cookie with multiple purposes can make sense. For instance, a cookie may be simultaneously used to enable a service, but also to track the user. A commonly encountered example for this are social media frames embedded in other websites.

However, some combinations of purposes are contradictory. The *ePrivacy Directive* [21] states that cookies which are strictly necessary for the operations of the site do not require consent. At the same time, Recital 30 of the GDPR states that unique identifiers may constitute personally identifying data, and hence require consent to be set in the visitor's browser. The contradiction thus occurs when a cookie is declared simultaneously as "Strictly Necessary", as well as "Analytics" or "Advertising" by the same website. Moreover, it is unclear how a consent notice will behave in general if multiple purposes are declared for the same cookie, and only one purpose is accepted.

This practice is relatively rare compared to the other observed potential violations. In our dataset, 926 out of 26 403 websites, i.e., 3.51%, declare multiple purposes for at least one cookie. Out of these, 269 websites declare a cookie as both "Strictly Necessary" and at least one additional category. In total we found 1555 cookies with multiple categories from the same websites, of which 346 were declared as "Strictly Necessary".

Note that for the training data extraction, all such cookies with multiple labels from the same website were removed prior to training the classifier.

7.7 Additional Analyses

In this section we will discuss two additional approaches to detect potential GDPR violations which expand on the ideas originally proposed by Nouwens [44] and Matte [38]. They require minor alterations to the crawler described in Chapter 3, namely through changing settings in the included Consent-O-Matic browser extension.

The extension automatically interacts with the Cookiebot, OneTrust and Termly consent notices in order to accept or reject consent for individual categories. In the context of the consent webcrawler, it is used to accept all categories of cookies to retrieve as much data as possible.

For the following analyses, we will change this configuration. For the first, we disable Consent-O-Matic entirely, such that no interaction occurs. For the second, we set the configuration so that all purpose categories (except "Strictly Necessary") are rejected, rather than accepted. This requires two additional crawls of the selected domains, which, due to time constraints, could not yet be performed in practice. However, in the following, we will provide a description of how the analyses can be performed in theory.

7.7.1 Detecting Implicit Consent

Recital 32 of the GDPR [22] specifies that consent must be given explicitly, namely as an action that is separate from the activity the user is currently pursuing. This is supported by the court decision on the Planet49 case [30] as well as the guidelines set out by the Article 29 Working Party [2]. Therefore, consent cannot be assumed implicitly, for example through continued use of the website, and requires the visitor to interact with the consent dialogue in order to be recognized as valid. Any non-essential cookies that are set before the notice is interacted with may thus be in violation of the GDPR.

Actors that set cookies without first obtaining consent have historically been fined for such actions. For instance, in December 2020, the French National Commission on Informatics and Liberty (CNIL) fined Google for breaching the French Data Protection Act, as they set advertising cookies on the user's browser without first acquiring proper consent, nor informing the user of their purpose [43]. Such types of violations hence build a strong case for potential legal action. In previous work by Nouwens et al., the authors analyzed the code of several consent notices, and identified various actions that websites recognize as consent. They found that 32.5% out of 680 crawled domains used some form of implied consent.

Using the consent label crawler, we can verify this observation in a different manner. The idea is to apply the consent label crawler as described in Section 3.3.2, but to change its configuration such that consent notices are never interacted with. By doing so, we can gather a dataset of cookies with associated cookie labels that have been set before any explicit consent has been provided by the visitor. Note that this does not impede the gathering of cookie labels, as this information can be accessed whether or not the consent notice has been interacted with.

The specific advantage of this approach, as compared to a regular web crawler that gathers cookies, is that we can distinguish cookies that are

strictly necessary for the operations of the website. As per the *ePrivacy Directive*, these cookies are the only category that does not require explicit consent, and can thus be set as soon as the visitor reaches the landing page of the website. Therefore, by filtering cookies with the "Strictly Necessary" type from the collected dataset, we receive a set of real observed cookies that per the website's own privacy policy should not have been set in the visitor's browser.

Through this approach, we can in theory gather evidence for implicit consent from the cookies set in the browser, which would be particularly helpful in determining potential GDPR violations.

7.7.2 Ignored User Consent Choices

Matte et al. [38] find that certain CMPs, particularly ones that are registered as part of the *Transparency and Consent Framework (TCF)* [26], may not actually respect the choices made by users. They find that 27 out of 1426 examined websites (1.89%) register positive consent even if the user has explicitly opted out. This is done through analyzing the inner workings of the TCF, which many Consent Management Platforms and advertisers use as a basis to store and exchange consent collected from visitors.

Through the use of our crawler and the Consent-O-Matic extension, we define an alternate method to detect whether a website respects the user's consent choices. By configuring Consent-O-Matic such that all cookie purposes (except "Strictly Necessary") are explicitly rejected, we can gather cookies that ignore the user's choices. Like in the previous approach, we again need to filter cookies that are declared as "Strictly Necessary", because these cookies do not require consent. All remaining cookies that belong to different purposes should not have been set, as they were explicitly rejected by the crawler. They hence represent evidence that the website in question ignores the user's choices and is non-compliant with the GDPR.

Note that for this approach to be sound, one needs to verify that the Consent-O-Matic extension is able to provide negative consent on all variants of the consent notices encountered in the wild. As the design of cookie banners and cookie walls can differ greatly between individual domains, and because they may be updated and changed over time, this can potentially be a time-intensive task, and must be performed prior to performing the crawl.

Chapter 8

Related Work

In this chapter we will describe some related work in the area of enforcing user privacy preferences and in the area of violation detection, some of which closely relate to the contributions we present in this thesis report.

8.1 User Privacy Enforcement

In this section we give an overview of some related approaches in enforcing user privacy online, including proposed standards such as the "*Do Not Track*" header or P3P, and compare them to CookieBlock.

8.1.1 Blocking of Third-Party Cookies

A standard defense mechanism against tracking and the collection of personal data is to block all third-party cookies. While this approach does not rely on outside support, it runs the risk of breaking website functionality, such as social media widgets, *Single Sign-On (SSO)* authentication, and other legitimate uses. Also, unlike CookieBlock, this mechanism cannot provide any protection against personal data collection from first-party cookies, such as *Google Analytics* cookies.

More recently, Firefox has introduced enhanced tracking protection, with more fine-grained controls such as the blocking of social-media tracker cookies, or cookies from unvisited websites.[1] However, there are as of yet no controls or tools that can reject cookies by individual usage purpose, a niche which CookieBlock can fill.

[1]See also the following Mozilla article: https://blog.mozilla.org/blog/2019/06/04/firefox-now-available-with-enhanced-tracking-protection-by-default/(2021.03.16)

8.1.2 "Do Not Track" Header

The "*Do Not Track*" header is a proposed W3C standard that defines a simple HTTP mechanism to allow users to express their desire not to be tracked [48]. The idea is that the user can enable the "*Do Not Track*" header in the browser settings, and the visited websites would then comply with this request, thus disabling their tracking technologies.

However, while the proposed standard is implemented by all major browser vendors, there is little incentive for website hosts to honor the request. This is because ignoring it comes with no potential legal consequences, and honoring it is thus entirely a matter of goodwill. Additionally, by January 2019, the working group for the proposed standard was closed due to the lack of support by third party hosts [23].

8.1.3 P3P

The Platform for Privacy Preferences (P3P) was a proposed W3C standard originally published in April 2002, which specified a computer-readable format for privacy policies [14]. Its intention was that such policies could be published by website hosts in a consistent format, which was to be automatically read and interpreted by a browser agent. Thus, the standard would save users the time required to read complex privacy statements. Previous studies, such as for example the works by McDonald et al. [39] and Jensen et al. [29], have shown that privacy policies are usually too time-consuming for users to read, and in many cases require university-level reading skills to be understood fully.

Much like the extension outlined in this report, the P3P standard used a set of user-specified privacy preferences to automatically accept or reject cookies and other browser tracking technologies. The main issue with the standard however was that it required both support from browser vendors as well as cooperation from website owners to implement the standard on their websites in a honest fashion. While browsers at the time, including Netscape Navigator 7 and Internet Explorer 6 up to 9 implemented P3P functionality, it never found wide adoption among website owners. Just like with the DNT header, the ostensible reason for this is that there was no incentive for hosts to implement the rather complicated protocol. As the GDPR did not exist at the time, there was no benefit to implementing it, and there was no threat of potential fines from regulators for not following the standard.

In fact, some high profile actors, including Google and Facebook, even bypassed the P3P controls implemented in Internet Explorer 9, exploiting a bug that made the browser accept all cookies regardless of the user's preferences [7]. Thus, with website owners refusing to implement the protocol,

browser vendors eventually dropped support for P3P as well. Work on the specification was discontinued and the document was retired by the W3C in August 2018.

What sets our approach to enforcing user's consent choices apart from the P3P and from the DNT header is that we do not need to trust advertisers or website owners to honor a protocol or correctly specify their usage purposes. Instead, we rely only on the accuracy of the trained model, and accept or reject cookies based on its predictions. While the classification will not have a perfect accuracy in all cases, it will also not run the risk of being abandoned due to a lack of outside support.

8.1.4 Automatic Interaction with Consent Notices

Legislation such as the *ePrivacy Directive* and the *GDPR* have lead to a sizeable increase in the number of cookie banners, walls and popups that a user encounters while browsing the web [16]. The frequency at which consent notices appear quickly became a nuisance to users [9]. With the resulting fatigue, users simply click the most visible button available to get rid of the consent notice, oftentimes being unaware of the consent they give through doing so [1, 4]. Cookie consent notices hence have become somewhat of a privacy concern in and of themselves, as users' desire to remove the privacy notice as fast as possible makes dark patterns a very effective way of retrieving positive consent from users [6].

A solution to this problem is offered by the Consent-O-Matic [28] and Cliqz Autoconsent [36] browser extensions. The idea is for the user to specify his consent preferences once at setup, which will then automatically be applied for all consent notices the user encounters. This automatically handles interaction with the consent notice, thus removing the hassle of having to deal with the banner or popup.

On the other hand, these tools also rely on outside support, in the sense that a consent notice needs to be implemented on the website in order for the extension to ensure that consent can be given or rejected. Moreover, they need to trust the website and the CMP to respect their choices, which as Matte et al. [38] showed is not always the case.

These extensions form a synergy with CookieBlock, where one works complementary to the other. Consent-O-Matic or Cliqz Autoconsent automatically select the user's consent choices, thus closing the consent notice and handling the nuisance, which CookieBlock itself is not capable of. CookieBlock on the other hand can provide an added layer of defense for websites that do not implement a consent notice, or which do not respect the user's choices. Even if a cookie is still set after rejection of a usage purpose, CookieBlock will then delete it automatically.

8.2 Detection of Potential GDPR Violations

In this section we provide a small overview of related work in the area of analyzing and verifying cookies compliance with the GDPR.

In a work by **Santos et al.** [47], the authors perform an extensive analysis of relevant binding and non-binding legislative documents that directly relate to the implementation of consent notices, including the GDPR, the ePrivacy Directive, and decisions from individual jurisdictions. They define 22 legal requirements for how a cookie banner should be designed, referencing both legal sources as well as domain expertise of the authors.

In addition to this, they determine how each requirement could be verified in practice, either through manual, semi-automated or fully automated analysis, or whether an extensive user study is needed. In summary, they find that most requirements do not allow for a fully automated analysis.

Nouwens et al. [44] analyse the emergence of dark patterns, and the prevalence of non-compliant consent notice designs on the web. They additionally identify how specific cookie banner designs affects the decisions visitors make. To test for non-compliant designs, they define three conditions for compliance:

1. Consent must be a clear and affirmative act, i.e., explicit.
2. Accepting all consent purposes must be just as easy as rejecting all options. The ”reject” option should moreover not be hidden.
3. No pre-selected consent options must exist, i.e., the consent must be opt-in, and not opt-out.

By scraping the design of five popular CMPs from the top 10k websites in the UK, they find that only 11.8% of 680 examined websites fulfill all three conditions. Furthermore, they find that removing the opt-out button increases positive consent by 22 percent, and that providing more granular consent controls reduces the affirmative consent by 8 to 22 percent. In our work, we expand on the analysis of implicit consent by proposing a different method to gather evidence for this criterion.

Utz et al. [55] set out to determine common properties in the design of consent notices, and they perform a large user study to identify how the position of the notice, range of choices and the framing of the content affects the decisions users make. They find that users are more likely to interact with notices in the lower left corner of the browser window, and that visitors are more likely to accept cookies if only a binary choice is given, rather than multiple opt-in choices of purposes.

Moreover, they find that 57.4% of 1000 examined domains utilize dark patterns to coax users into giving positive consent. This is most commonly

done by highlighting the "*Accept All*" button, or hiding the option to reject consent for individual usage purposes. Furthermore, they show that the dark pattern of nudging increases the acceptance rate for mobile and desktop users by roughly 11% resp. 6%.

Matte et al. [38] study *IAB Europe's Transport and Control Framework (TCF)*, and using its specifications, they are able to identify several potential legal violations in CMP behavior. They analyse the storage of consent of 1426 individual websites, and using two automatic, and one semi-automatic web crawl, they find that:

- 141 websites send positive consent without the user having made a choice.
- 236 websites use pre-selected options in their consent notice dialogues.
- 27 websites store positive consent after the user has explicitly opted out.

Similar to the work presented in this master thesis, they provide a browser extension called Cookie Glasses, which enables users to verify whether CMPs that are part of the TCF respect their consent choices.

In our work, we also propose a crawl that can, through the analysis of collected cookies and the rejection of consent, verify whether the user's choices are respected.

Chapter 9

Conclusion

In this master thesis report, we presented a novel approach to enforcing user privacy preferences in the domain of cookies, and we implemented this in the form of a browser extension called CookieBlock.

By detecting selected Consent Management Platform implementations in a large set of 1.6 million domains, we scraped cookies and corresponding consent labels from a subset of roughly 26 400 websites. In total we obtained a dataset of over 300 000 cookie samples, which were labeled by multiple different sources, including large repositories and individual hosts.

After having collected this dataset, we devised a list of feature extraction steps to extract numerical data from cookies, which was then be provided as input to a series of tree ensemble classifiers.

Using popular algorithms such as XGBoost, we received a predictor that manages to outperform the manually classified *Cookiepedia* repository with an overall accuracy of 87.2%, with improved precision and recall in most categories. To the best of our knowledge, we are the first to apply machine-learning techniques to the domain of cookies.

This predictor was then integrated as the core component of a browser extension we titled CookieBlock, which allows the user to specify their cookie consent preferences. All cookies that serve purposes the user did not agree to are subsequently deleted by the extension.

With the use of the collected dataset, we were also able to observe evidence for potential GDPR violations. By drawing from knowledge of specific articles of the GDPR and related legislation, we devised six novel methods which researchers and authorities can use to identify non-compliance on websites. Only 8.13% of all 26 403 websites we analyzed did not produce any evidence for potential GDPR violations, suggesting that most hosts still have trouble fulfilling the requirements of the GDPR.

Bibliography

[1] Alessandro Acquisti and Jens Grossklags. Privacy and rationality in individual decision making. *Security & Privacy, IEEE*, 3:26 – 33, 02 2005.

[2] Article 29 Working Party, European Union. Guidelines on Consent under Regulation 2016/679 (WP259 rev.01), 2018. https://ec.europa.eu/newsroom/article29/item-detail.cfm?item_id=623051; Last accessed on: 2021.02.06.

[3] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[4] Rainer Böhme and Stefan Köpsell. *Trained to Accept? A Field Experiment on Consent Dialogs*, page 2403–2406. Association for Computing Machinery, New York, NY, USA, 2010.

[5] Elena Boldyreva. Cambridge analytica: Ethics and online manipulation with decision-making process. pages 91–102, 12 2018.

[6] C. Bösch, B. Erb, F. Kargl, Henning Kopp, and Stefan Pfattheicher. Tales from the dark side: Privacy dark strategies and privacy dark patterns. *Proceedings on Privacy Enhancing Technologies*, 2016:237 – 254, 2016.

[7] Jon Brodkin. Google tricks Internet Explorer into accepting tracking cookies, Microsoft claims, February 2012. https://bit.ly/3exMVMp; Accessed on 13.03.2021.

[8] BuiltWith.com. Privacy compliance usage distribution in the top 1 million sites, October 2020. https://web.archive.org/web/20201021075918/https://trends.builtwith.com/widgets/privacy-compliance/.

[9] Matt Burgess. The tyranny of GDPR popups and the websites failing to adapt. https://www.wired.co.uk/article/gdpr-cookies-eprivacy-regulation-popups, August 2018. Accessed on 2021.02.07.

[10] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

[11] Commission Nationale de l'Informatique et des Libertés (CNIL). Délibération n 2019-093 du 4 juillet 2019 portant adoption de lignes directrices relatives à l'application de l'article 82 de la loi du 6 janvier 1978 modifiée aux opérations de lecture ou écriture dans le terminal d'un utilisateur (notamment aux cookies et autres traceurs), July 2019.

[12] Intersoft Consulting. GDPR Fines/Penalties. https://gdpr-info.eu/issues/fines-penalties/. Last accessed on: 2021.02.06.

[13] CookiePro. Legacy article - categorizing cookies. https://web.archive.org/web/20210208155826/https://community.cookiepro.com/s/article/UUID-6f01b88c-0440-0642-3610-819c6ca0f7c4, Jan 2021. Accessed on: 2021.02.08.

[14] Lorrie Faith Cranor. P3P: Making privacy policies more useful. *IEEE Security and Privacy*, 1(6):50–55, November 2003. https://www.w3.org/TR/P3P11/.

[15] Adrian Dabrowski, Georg Merzdovnik, Johanna Ullrich, Gerald Sendera, and Edgar Weippl. *Measuring Cookies and Web Privacy in a Post-GDPR World: Methods and Protocols*, pages 258–270. 03 2019.

[16] Martin Degeling, Christine Utz, Christopher Lentzsch, Henry Hosseini, Florian Schaub, and Thorsten Holz. We value your privacy ... now take some cookies: Measuring the gdpr's impact on web privacy. *CoRR*, abs/1808.05096, 2018.

[17] Anna Veronika Dorogush, Andrey Gulin, Gleb Gusev, Nikita Kazeev, Liudmila Ostroumova Prokhorenkova, and Aleksandr Vorobev. Fighting biases with dynamic boosting. *CoRR*, abs/1706.09516, 2017.

[18] R. V. Eijk, H. Asghari, Philipp Winter, and A. Narayanan. The impact of user location on cookie notices (inside and outside of the european union). 2019.

[19] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16,

page 1388–1401, New York, NY, USA, 2016. Association for Computing Machinery.

[20] European Parliament, Council of the European Union. Directive 95/46/ec of the european parliament and of the council)), October 1995. http://data.europa.eu/eli/dir/1995/46/oj; Last accessed on: 2021.03.13.

[21] European Parliament, Council of the European Union. Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications)), July 2002. http://data.europa.eu/eli/dir/2002/58/oj; Last accessed on: 2021.02.06.

[22] European Parliament, Council of the European Union. Regulation (EU) 2016/679 Of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), April 2016. http://data.europa.eu/eli/reg/2016/679/2016-05-04; Last accessed on: 2021.02.06.

[23] Github. "WG closed · w3c/dnt@5d85d6c". https://github.com/w3c/dnt/commit/5d85d6c3; Accessed on 2021.14.03.

[24] Jens Grossklags and Nathan Good. Empirical studies on software notices to inform policy makers and usability designers. In Sven Dietrich and Rachna Dhamija, editors, *Financial Cryptography and Data Security*, pages 341–355, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[25] Maximilian Hils, Daniel W. Woods, and Rainer Böhme. Measuring the emergence of consent management on the web. In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, page 317–332, New York, NY, USA, 2020. Association for Computing Machinery.

[26] Interactive Advertising Bureau (IAB) Europe. Transparency and Consent Framework - CMP List. https://iabeurope.eu/cmp-list/, February 2021. Accessed on 2021.02.08.

[27] International Chamber of Commerce UK. ICC UK Cookie guide, November 2012. https://www.cookielaw.org/wp-content/uploads/2019/12/icc_uk_cookiesguide_revnov.pdf; Accessed on 2021.02.08.

[28] Rolf Bagge Janus Bager Kristensen. Consent-O-Matic, 2020. https://github.com/cavi-au/Consent-O-Matic.

[29] Carlos Jensen and Colin Potts. Privacy policies as decision-making tools: An evaluation of online privacy notices. pages 471–478, 01 2004.

[30] Judgement of the Court (Grand Chamber. Case C-673/17 Planet49 GmbH v Bundesverband der Verbraucherzentralen und Verbraucherverbände – Verbraucherzentrale Bundesverband e.V. ECLI:EU:C:2019:246, 1 October 2019. http://curia.europa.eu/juris/document/document.jsf?docid=218462&doclang=EN; Last accessed on: 2021.02.06.

[31] Schawb K., Marcus A., Oyola J., Hoffman W., and Luzi M. Personal data: The emergence of a new asset class, 2011.

[32] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. 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.

[33] Oksana Kulyk, Annika Hilt, Nina Gerber, and Melanie Volkamer. "This Website Uses Cookies": Users' perceptions and reactions to the cookie disclaimer. 04 2018.

[34] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. "Tranco: A research-oriented top sites ranking hardened against manipulation". In *Proceedings of the 26th Annual Network and Distributed System Security Symposium*, NDSS 2019, February 2019.

[35] Timothy Libert, L. Graves, and R. Nielsen. Changes in third-party content on european news websites after GDPR. 2018.

[36] Sam Macbeth. Cliqz Autoconsent, December 2020. https://github.com/cliqz-oss/autoconsent.

[37] Dominique Machuletz and Rainer Böhme. Multiple Purposes, Multiple Problems: A user study of consent dialogs after GDPR. *CoRR*, abs/1908.10048, 2019.

[38] C. Matte, N. Bielova, and C. Santos. Do cookie banners respect my choice? Measuring legal compliance of banners from IAB Europe's Transparency and Consent Framework. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 791–809, 2020.

[39] A. M. McDonald and L. Cranor. The cost of reading privacy policies. 2009.

[40] Lynette I. Millett, Batya Friedman, and Edward Felten. Cookies and web browser design: Toward realizing informed consent online. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, page 46–52, New York, NY, USA, 2001. Association for Computing Machinery.

[41] Mozilla. MDN Web Docs – using HTTP cookies. https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies. Accessed on: 2021.02.07.

[42] Mozilla. OpenWPM – a web privacy measurement framework. https://github.com/mozilla/OpenWPM, 2020. Version used: 0.12.0.

[43] Commission nationale de l'informatique et des libertés (CNIL). Cookies: financial penalties of 60 million euros against the company GOOGLE LLC and of 40 million euros against the company GOOGLE IRELAND LIMITED, December 2020. https://bit.ly/3rFIjaF; Accessed on: 2021.02.07.

[44] Midas Nouwens, Ilaria Liccardi, Michael Veale, David Karger, and Lalana Kagal. Dark patterns after the GDPR: Scraping consent popups and demonstrating their influence. *CoRR*, abs/2001.02479, 2020.

[45] OneTrust. Cookiepedia. https://cookiepedia.co.uk/. Accessed on 2021.02.08.

[46] Iskander Sanchez-Rola, Matteo Dell'Amico, Platon Kotzias, Davide Balzarotti, Leyla Bilge, Pierre-Antoine Vervier, and Igor Santos. "Can I Opt Out Yet?": GDPR and the global illusion of cookie control. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, Asia CCS '19, page 340–351, New York, NY, USA, 2019. Association for Computing Machinery.

[47] Cristiana Santos, Nataliia Bielova, and Célestin Matte. Are cookie banners indeed compliant with the law? Deciphering EU legal requirements on consent and technical means to verify compliance of cookie banners. *CoRR*, abs/1912.07144, 2019.

[48] D. Singer and R. Fielding. Tracking preference expression (DNT) W3C working group note. https://www.w3.org/TR/tracking-dnt/, January 2019.

[49] StatCounter. Desktop browser market share worldwide. https://web.archive.org/web/20210227093820/https://gs.statcounter.com/browser-market-share/desktop/worldwide, February 2021.

[50] CookieBot Support. Unclassified cookies - how do I classify them manually? https://web.archive.org/web/20201111204915/https://support.cookiebot.com/hc/en-us/articles/360003735214-Unclassified-cookies-how-do-I-classify-them-manually-, May 2018. Accessed on: 2021.02.08.

[51] David Temkin. Charting a course towards a more privacy-first web. https://web.archive.org/web/20210303152006/https://blog.google/products/ads-commerce/a-more-privacy-first-web/, March 2021.

[52] Martino Trevisan, Stefano Traverso, Eleonora Bassi, and Marco Mellia. 4 Years of EU Cookie Law: Results and Lessons Learned. *Proceedings on Privacy Enhancing Technologies*, 2019:126–145, 04 2019.

[53] Tobias Urban, Martin Degeling, Thorsten Holz, and Norbert Pohlmann. *Beyond the Front Page: Measuring Third Party Dynamics in the Field*, page 1275–1286. Association for Computing Machinery, New York, NY, USA, 2020.

[54] Tobias Urban, Dennis Tatang, Martin Degeling, Thorsten Holz, and Norbert Pohlmann. The Unwanted Sharing Economy: An analysis of cookie syncing and user transparency under GDPR. *CoRR*, abs/1811.08660, 2018.

[55] Christine Utz, Martin Degeling, Sascha Fahl, Florian Schaub, and Thorsten Holz. (Un)informed Consent: Studying GDPR Consent Notices in the Field. *CoRR*, abs/1909.02638, 2019.

[56] Daniel W Woods and Rainer Böhme. The commodification of consent. 05 2020.

Appendix A

Consent Crawler Details

In this appendix, we describe how consent data is retrieved for the *Cookiebot*, *OneTrust* and partially also the *Termly* CMPs, although the latter did not provide many training data results.

A.1 Cookiebot Consent Crawler

The Cookiebot consent notice (see Figure A.1) stores the description and label for each cookie inside a JavaScript document called `cc.js`. To access this document, it is first required to find a website-specific UUID string in the landing page `HTML`, using which one can access the file on the remote Cookiebot domain. This string can either be found as an attribute called `"cbid"` as part of a `href` HTML tag, or directly contained as part of a complete URL related to the Cookiebot domain.

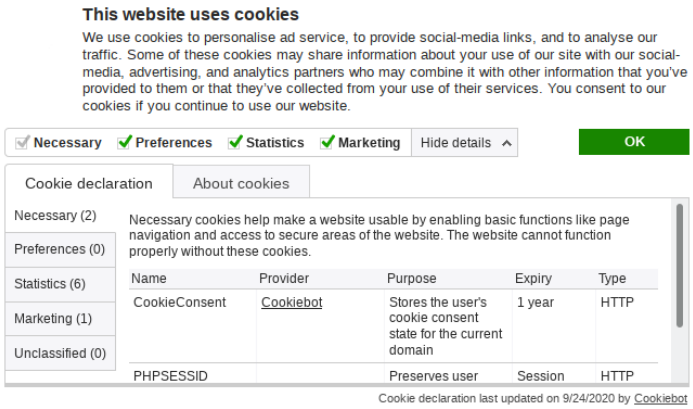

Figure A.1: Typical example of the Cookiebot consent notice. Its design is fairly consistent across the majority of websites that make use of it.

```
CookieConsentDialog.cookieTableNecessary    = [[..], [..], ...];
CookieConsentDialog.cookieTablePreference   = [[..], [..], ...];
CookieConsentDialog.cookieTableStatistics   = [[..], [..], ...];
CookieConsentDialog.cookieTableAdvertising  = [[..], [..], ...];
CookieConsentDialog.cookieTableUnclassified = [[..], [..], ...];
```

Figure A.2: The JavaScript arrays in which Cookiebot stores all cookie descriptions. Each array corresponds to a consent label.

Then, one needs to access the following URL:

`https://consent.cookiebot.com/<UUID>/cc.js`

And inside `cc.js`, we find a set of inline arrays as shown in Figure A.2. Each array lists all cookies with descriptions and expiration date for the category corresponding to the name of the array. The "Unclassified" array specifically lists cookies that the website host neglected to assign to a category. These are ignored for the purpose of training a classifier, while the other categories can be assigned exactly to our internal representation.

Note that for Cookiebot, it is necessary to specify the domain of the website that uses the CMP as the referrer in the GET header when accessing `cc.js`. Otherwise, the CMP will refuse to display the document. The same occurs if one tries to connect from a geographical region in which the notice is not supposed to be displayed. This is another reason why using a VPN from within a EU member state is required.

Cookiebot records *Name*, *Provider*, *Purpose*, *Expiry* and *Type* of the cookie. The *provider* is a list of domains, one of which will be assigned to the actual cookie set in the browser. The *purpose* is a description of what the cookie will be used for. The *expiry* is a coarse indication of the retention period of the cookie, usually given in months, years or some other time interval. This field may also contain the text `"Session"` if it is a session cookie.

The "*Type*" field indicates what kind of entry is declared. Cookiebot does not only store cookie declarations, but also other types of browser tracking technologies. Apart from cookies that are set through HTTP requests or JavaScript, the most common declared type here are *Tracking Pixels*, which are small 1x1 pixel images embedded in a site that are retrieved from an advertiser domain. For the purpose of training data extraction, and for the analyses performed in Chapter 7, any non-cookie entries are ignored.

A.2 OneTrust Consent Crawler

The OneTrust consent notice, as shown in Figure A.3, is generally implemented in one of two different fashions, and uses one of the domains listed in Figure 3.2.

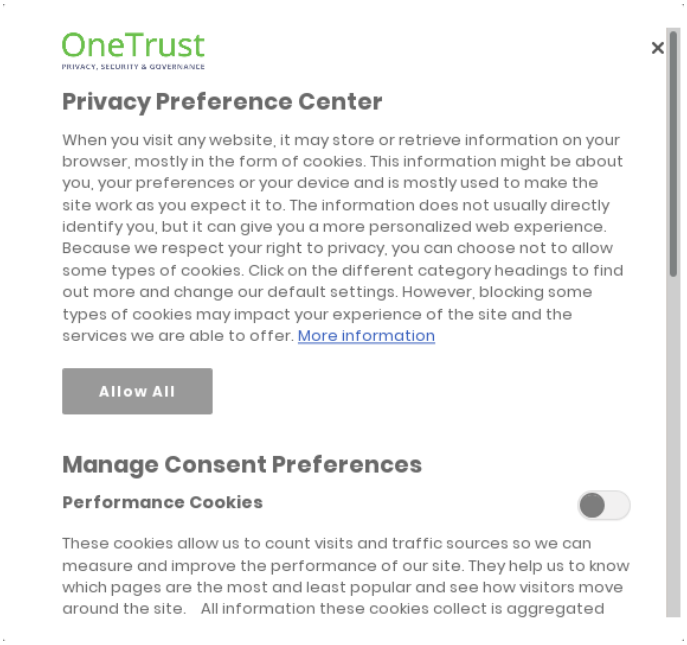

Figure A.3: One possible appearance of the OneTrust consent notice. Note that the design is generally very variable. Some instances may also include an OptAnon or CookiePro logo.

1. In the first case, the cookie label data is stored inside a series of JSON documents, accessed through a collection of region-specific rulesets. For this variant, the crawler first needs to find a HTML script tag attribute called `"data-domain-script"`, which contains the UUID required for accessing the rulesets.

   The rulesets are located at the path:

   `<OT_URL>/consent/<DD_UUID>/<DD_UUID>.json`

   Where `<OT_URL>` is the corresponding OneTrust URL that was detected on the landing page of the targetted website. This is usually one of the domains listed in Figure 3.2.

   Each ruleset is a JSON document that contains language- and website-specific identifiers, which in a second step can be used to retrieve the actual cookie label information from a second JSON document. Due to language barriers, we only retrieve English and German cookie information. The exact JSON document path is:

   `<OT_URL>/consent/<DD_UUID>/<CC_UUID>/<LANG>.json`

   Where `<CC_UUID>` is the identifier found in the ruleset JSON document, and `<LANG>` is any associated language string, for instance “en” or “de”, depending on which language is supported.

"Strictly Necessary"	"mandatory" "necessary"	"essential" "required"	
"Functional"	"functional" "preference"	"security" "secure"	"video" "social"
"Performance/Analytics"	"analytic" "statistic"	"anonymous" "research"	"performance" "measurement"
"Tracking/Advertising"	"advertise*" "ad selection" "sale of data" "fingerprint"	"targeting" "personalize" "tracking" "geolocation"	"personalization" "marketing" "tracker" "personal info"

Figure A.4: Mapping of keywords to category. The left-hand column displays the internal categories, while the right-hand column contains all keywords that, if found inside a OneTrust category name, map the cookie to that internal category.

2. The second common implementation of the OneTrust CMP foregoes the rulesets and instead directly includes the cookie-to-category assignments, along with all purpose details, inside a single JavaScript document. For instance, for the cookielaw domain, it can be found at:

   `https://cdn.cookielaw.org/consent/<uuid_pattern><name>.js`

For both cases, the cookie labels and associated metadata is stored in a JavaScript sub-object, starting with the string key `"Groups"`. Cookies are listed under each vendor domain, and each entry stores the assigned consent category as a string. Unfortunately, no unambiguous category ID value appears to exist, and websites can define their own arbitrary, language-specific category names.

For the purpose of classifying cookies, we use a fixed set of four distinct categories, including “Strictly Necessary”, “Functionality”, “Analytics” and “Advertising”, as described in Section 3.2. As such, in order to map the OneTrust group names to the internal categories, we define a set of characteristic keywords. If a keyword is detected in a group name, the cookie is mapped to the corresponding internal category. The full list of keywords is presented in Figure A.4.

We also utilize an equivalent mapping for German category names. If multiple terms are present in the website, the least privacy preserving category takes precedence.

In terms of content, OneTrust lists the *name* and *host*, *purpose*, *expiry*, and whether the cookie is a *session* cookie or *persistent*. The expiry is in this case always listed in days, unless we have a session cookie.

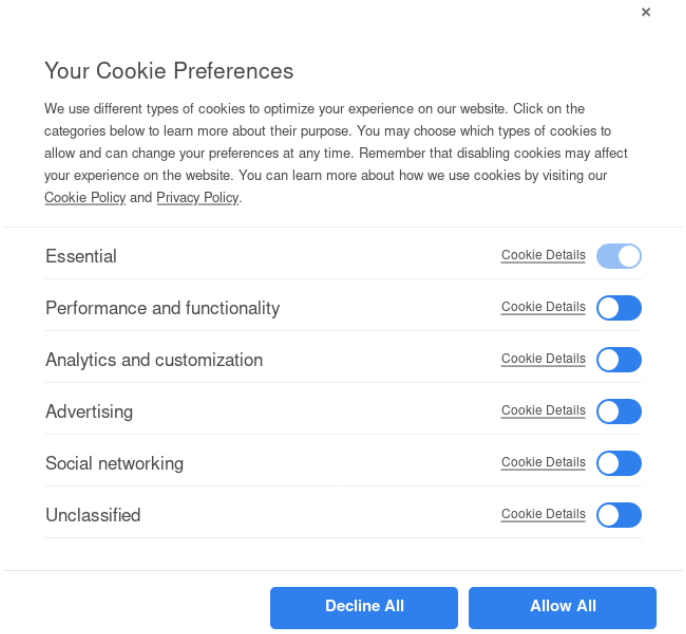

Figure A.5: Example design of the Termly consent notice, displaying all available category types.

A.3 Termly Crawler

The Termly CMP is an example of a Consent Management Platform that is not part of IAB Europe's TCF [26]. Its interface usually defines the same four categories that we utilize internally for classification, with the addition of two categories called "Social Networking" and "Unclassifed".

For the implementation of our classifier, we considered adding the category of *"Social Media Cookies"* as a potential fifth option. One particular reason for this is that cookies belonging to social media websites are often split between offering functionality as well as tracking the user [45].

On the one hand, they provide functionality, usually in the form of a embedded widget or frame that allows users to use features of social media, such as liking a page or sharing it with others. On the other hand, websites such as *Facebook* are notorious for collecting personal data and tracking users. Such embedded frames are hence also used for the purpose of tracking visitors across different sites. As such, these cookies do not fully align with either category

Our main reason for ultimately not including the category was a lack of training samples. Despite the category being observed for both the Termly and OneTrust CMPs, there was too little data to properly train the classifier.

The “Social Networking” category is hence treated the same as unclassified cookies, and is excluded for training.

For the Termly CMP, we extract consent data similarly to the OneTrust and Cookiebot. First, it is required to find a UUID string that is stored inside a HTML script tag on the websites landing page. This allows us to access a JSON document at the following address:

```
"https://app.termly.io/api/v1/snippets/websites/<uuid1>"
```

This JSON document contains the cookie policy. By parsing its contents, one finds another UUID pointing to the document storing all cookie purpose information. This can be found at the following path of the same site:

```
"/api/v1/snippets/websites/<uuid1>/documents/<uuid2>/cookies"
```

This is a JSON document that stores all relevant cookie consent data. Termly stores the *name*, *domain*, *purpose*, *expiry*, *type of tracker* and even the cookie's *value* in its consent notice. Unfortunately, due to the low number of training samples we managed to collect for this CMP, and because most of the declarations were uncategorized, this implementation is ultimately not very useful for classifier training or violation detection.

A.4 Other Target CMPs

In this subsection we refer to Table 3.1. Apart from *Cookiebot*, *OneTrust* and *Termly*, we additionally have the CMPs *Cookie Script* and *Cookie Information* that could serve as future crawl targets. Note that *OptAnon* and *CookiePro* are included as part of the OneTrust crawl, as they were both acquired by the latter, and use the same internal structure.

Another interesting Consent Management Platform to target is *Borlabs Cookie*. Despite not being remotely hosted, Borlabs usually displays a uniformly structured consent notice with cookie labels. This may be very useful because, according to *BuiltWith*, Borlabs enjoys significant popularity particularly in Germany at 10.76% coverage out of roughly 800 000 websites, thus offering many potential crawl targets [8].

Appendix B

Feature Engineering Details

In Section 4.2 we gave an overview of the feature engineering steps that are applied to cookies to transform them into numerical feature vectors. Many of the features listed in Tables 4.1, 4.2 and 4.3 are self-explanatory, but some require more elaboration. As such, this appendix will describe a select number of features in more detail.

B.1 Name and Domain Features

The "*Top Names*" and "*Top Domains*" features are one-hot vectors which indicate whether a cookie has a particular name or domain. As we cannot represent every single name in a feature, we restrict ourselves to only the top 100 most common names and domains. More specifically, the kth entry of the vector indicates whether the cookie's name, or respectively the cookie's domain exactly matches the kth most common name resp. domain sourced from an external ranking.

These rankings are hereby constructed from a secondary dataset of cookies obtained from crawling random websites. We pick random domains so that the resulting ranking is not biased on the cookies that only appear on sites that use OneTrust or Cookiebot CMPs. From the collected data, we count the number of occurrences of each cookie per website, and sort them to receive the rankings we need. Some of the most common cookie names include the Google Analytics cookies such as `"_ga"` and `"_gid"`, or identifier cookies such as `"uid"` or `"NID"`. Among the most common domains are advertisers such as `"doubleclick"` and `"pubmatic"`.

These features are hence similar to a bag-of-words approach, and should allow the classifier to distinguish the types of common first-party cookies, or third-party cookies originating from a select number of advertisers.

B.1.1 Name Patterns

In addition to cookies whose names match exactly between hosts, there also exist certain types of cookies that include a pseudo-randomly generated unique identifier within their name. Such cookies follow a pattern in that, if one were to remove the unique identifier, they would be identical across all domains. A common example for this is again Google Analytics, which has a cookie with the following regular expression pattern:

```
^_gat_gtag_UA_[0-9]+_[0-9]+$
```

Figure B.1: Example of a pattern cookie, belonging to the Google Analytics family.

To be able to also identify pattern cookies, we performed a semi-automated analysis of the collected dataset, by first sorting all cookies names that have been encountered in alphabetical order, and then selecting all names that occur in a sufficiently long contiguous sequence, where at least 4 characters match exactly. This filters a candidate list of potential pattern type cookies, which in a second step need to be analyzed manually by a human operator in order to construct a regular expression to represent them, as above.

In this fashion, we constructed a ranking of more than 50 pattern cookies, sorted by occurrence in a randomly collected dataset. This ranking is then used to construct another one-hot vector feature which identifies common pattern names found in the wild.

B.1.2 Name and Content Tokens

The name, as well as the content of a cookie may contain human-readable English tokens. While they are not be separated by whitespaces, it is still possible to identify frequent tokens by looking up substrings of the name and the cookie content in a dictionary. Common terms found inside cookies include `"consent"`, `"landing"`, `"Id"`, `"timestamp"`, `"version"`, `"site"`, `"test"` and `"user"`, among others.

For those substrings for which we find an entry in the dictionary, we again construct a ranking of the most common occurrences as before, and use this to construct an indicator vector feature for both the tokens in cookie names and cookie contents.

B.2 IAB Vendor Feature

The *Transparency and Consent Framework (TCF)* is a set of technological specifications defined by IAB Europe which describes how a CMP can interact with third-party vendors and website hosts in order to transmit consent and

to revoke it at the proper time. Those vendors that are part of this TCF are listed on IAB's website, which could be a valuable indicator as to which domains are potential advertisers, or which offer web-analytics services [26].

As such, we scraped the TCF website to retrieve a complete set of domains that are part of the TCF's vendor list. The feature is then defined as a binary indicator. For each cookie, we set the indicator to 1 if the listed cookie domain is in the set of TCF vendors, and 0 otherwise.

B.3 Expiration Time Features

The expiration time can tell us much about how a cookie is used. A session cookie for instance is likely to not be used for the purpose of tracking, as they are deleted when the browser is closed. The same applies to persistent cookies that last only for a few seconds.

To identify such nuances, we apply a range of different features, including an ordinal feature that tracks the expiration time in seconds, a binary indicator feature that is true when the cookie is a session cookie, as well as a one-hot vector of certain expiration time intervals.

The idea of the latter is to provide more context to the classifier which a single ordinal feature could not provide. We include the following time intervals as separate feature indicators:

- Less than 1 hour
- Between 1 and 12 hours
- Between 12 hours and 1 day
- Between 1 day and 1 week
- Between 1 week and 1 month
- Between 1 and 6 months
- Between 6 and 18 months
- More than 18 months

B.4 Unique Identifier Features

Unique identifiers are commonly used as part of tracking cookies, however, with a broad range of methods to generate such identifiers, it is not easy to detect whether a cookie contains such a value in its contents.

B.4.1 Randomly Generated Strings

One common approach to create unique identifiers is to use pseudo-random string generation. These strings usually enjoy high entropy, while natural language terms or repetitive numbers that are not used for identifying users usually enjoy low entropy. Therefore, one approach to detect potential unique identifiers is to create features that reflect the entropy of the cookie's content.

As a heuristic, we use the *zlib* library to compress the value of a cookie and then compare the size of the compressed content to the original size. This is done because high entropy data usually cannot be compressed very well. For a more direct approach, we also compute the *Shannon entropy* for the same purpose.

B.4.2 Universal Unique Identifiers (UUID)

The canonical universal unique identifier (UUID) is a sequence of hexadecimal numbers that follows a fixed format. The way in which it is generated is determined by the UUID version.[1]

We extract the version number in the hopes that it can help identify hidden patterns in cookies. The UUID versions are as follows:

- Version 1: Uses current time and MAC address of machine.
- Version 2: Rarely seen in practice, a DCE Security UUID, see RFC 4122.
- Version 3: Uses predefined identifiers, hashed using MD5.
- Version 4: Pseudo-randomly generated hexadecimal strings.
- Version 5: Uses predefined identifiers, hashed using SHA-1.

B.4.3 Timestamp and Dates

Timestamps and dates are another common method through which a cookie can uniquely identify users, as the current timestamp at which the user first accesses a website is likely to be unique. Thus they can facilitate tracking. At the same time, date strings may also be used in functional cases.

With these features, we identify content in cookies that is formatted like a human-readable date, e.g., `"2021-03-14"`, or `"1st August 2019"`, or like a UNIX timestamp , e.g., `1615725910`.

[1] See also: https://www.ietf.org/rfc/rfc4122.txt

B.5 Locale Content Strings

We observed numerous cookies which contain strings that specify some type of locale content, including country, currency, language or even keyboard layout. We expect this information to serve mostly functional purposes, for instance to identify the language in which a website needs to be displayed, the keyboard layout being used, the country the user is connecting from to display a flag icon, etc.

As such, we added an indicator feature which identfies whether a cookie contains some form of locale content. A large dictionary of potential identifiers is obtained using the *pyenchant* Python package, and we use it to identify whether the cookie content matches such a locale string. A similar method is used for the JavaScript variant of the feature extraction.

B.6 Content Encoding Features

The content of a cookie can often be encoded in the form of JSON objects, or formatted such that several values are split by a certain separator string. For JSON, we parse the entire content as a JSON object, count the number of entries in the object as well as the types that are used. The types include numbers, alphabetical content, timestamps, and subobjects.

For CSV data, we try to find an identifier that best fit to separate individual values in a cookie. Period and dashes are considered separately, as they are very common separators that deserve their own feature. We count the number of separations, as well as the contained types of data.

Finally, we also try to decode the cookie content as Base64. We do not have a surefire method to determine whether the content of a cookie really is Base64 encoded, hence we simply apply the heuristic to attempt the decoding, and if it happens to work, we will set the feature entry to 1, and 0 otherwise.

B.7 String Similarity Metrics

In order to determine how much a cookie changed between updates, we apply both the *Levenshtein* distance, as well as a string similarity metric based on *Gestalt Pattern Matching*, which is offered by the Python *difflib* library.

These features may be slow to compute, as the complexity of the corresponding algorithms is quadratic, and thus rises quickly for larger cookie contents. For the JavaScript implementation, we used a module that reimplements Python's *difflib* functions.

Appendix C

Repositories and Dataset

In the following, we provide links to repositories containing the code of each component presented in this report, and describe their contents. We also provide the collected cookie data and violation detection statistics.

C.1 Web Crawler Implementations

The code for the *CMP Presence Crawler*, as well as the *Cookie Consent Label crawler*, which we have described in Sections 3.3.1 and 3.3.2 respectively, can be found at the following GitHub repository:

- https://github.com/dibollinger/CookieBlock-Crawler-Prototype
- https://github.com/dibollinger/CookieBlock-Consent-Crawler

The first repository contains an earlier implementation of the consent label crawler that did not make use of the OpenWPM framework, but instead only used the Selenium library. It is a sequential crawl that only retrieves the cookie labels, but not the cookies themselves. The second contains both the presence and the consent label crawler as described in this report.

Each repository contains a README file that explains what each folder contains, and how to make use of the crawler scripts.

C.2 Classifier and Feature Extraction

The feature extraction scripts, as well as the classifier implementations written in Python, as described in Sections 4.2 and 4.3 respectively, can be found in the following repository:

- https://github.com/dibollinger/CookieBlock-Consent-Classifier

Instructions on how to extract the features, as well as how to train and evaluate each classifier model are given in the repository's README.

C.3 CookieBlock

The extension itself, as described in Chapter 5, can be found at the following repository:

- https://github.com/dibollinger/CookieBlock

Additionally, the repository also includes the reimplementation of the feature extraction using NodeJS, which was described in Section 5.3.1.

As of the time of writing, CookieBlock has not been released on any browser platforms yet, and only Firefox support is implemented. Support for different Chromium-based browsers, such as *Chrome* and *Edge*, will however be added in the future. To test the extension, we used the *web-ext* tool[1].

C.4 Violation Detection and Other Scripts

The Python scripts used to gather evidence for potential violations (as described in Chapter 7) can be found in the following repository:

- https://github.com/dibollinger/CookieBlock-Other-Scripts

This repository also contains all auxiliary scripts used for other sections of the report. This includes:

- The database processing scripts with corresponding SQL files.
- The scripts used to scrape data from Cookiepedia.
- The scripts used to build resources for the feature extraction.
- And the scripts used to generate statistics on the collected data.

Each folder in the repository contains a README that explains the code and its usage in greater detail.

C.5 Collected Datasets

The databases containing all the cookie and consent label data we collected can be found in the following Google Drive folder:

- https://drive.google.com/drive/folders/1P2ikGlnb3Kbb-FhxrGYUPvGpvHeHy5ao

This also includes the domain lists we targeted in our web crawls, the transformed training data samples in JSON format, the corresponding extracted features, the trained classifier models and performance evaluations, as well as the violation detection results. Please refer to the included README for more information.

[1] See: https://github.com/mozilla/web-ext

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Analyzing Cookies Compliance with the GDPR

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Bollinger	Dino Mario

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Steinhausen, 16.03.21	D. Bollinger

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.