Developing a Web-based Tool for Detecting Deceptive Designs in Cookie Banners

Braullo Jose A. Jo*[1], Shanea J. Olino[1],
Ligaya Leah Figueroa[2], Ma. Rowena C. Solamo[2], and Rommel P. Feria[2]

[1] University of the Philippines Diliman, Quezon City 1101, Philippines
[2] Web Science Group, University of the Philippines Diliman, Quezon City 1101, Philippines
braullojo.bj@gmail.com

Abstract. Deceptive designs, also known as dark patterns, are user interface tricks websites and applications use to manipulate user behavior and collect data without informed consent. These patterns include misleading language, asymmetrical options, and hidden information. Websites commonly manifest these deceptive designs in cookie banners to track user behavior. Given the sensitive nature of the data cookies may contain, it is crucial to flag and address potential manipulations of user consent through deceptive designs in cookie banners. This study builds on Ariadne, a browser extension developed by Adorna et al. that identifies deceptive patterns in cookie banners. To improve on their work, this study presents VeraCookie, a web application that has similar functionality but is compatible with all browsers and devices. This application integrates advanced machine learning models, including a Random Forest Classifier for assessing language clarity and a Vision Transformer (ViT) model for evaluating the symmetry of the options present in a cookie banner. Results show that VeraCookie outperforms the previous tool, achieving higher accuracy and better user experience. This study aims to demonstrate the effectiveness of VeraCookie in defense against deceptive designs in cookie banners.

Keywords: cookie banners, dark patterns, deceptive designs, machine learning, random forest classifier, vision transformer

1 INTRODUCTION

1.1 Background

Deceptive designs have become prevalent across various digital platforms, from e-commerce websites to social media. Deceptive designs, also known as dark patterns, are tricks websites and applications employ to push users into doing specific actions, primarily to generate more engagement and collect more data [3]. These practices raise ethical concerns and discussions about digital transparency. Not only do they reduce users' trust, but they also raise questions regarding the boundaries of acceptable user interaction. One common way for websites

J. Caro et al. (eds.), Proceedings of the Workshop on Computation: Theory and Practice (WCTP 2024) , Atlantis Highlights in Computer Sciences 23,
https:// doi.org/10.2991/978-94-6463-684-0_6

to employ deceptive designs is through cookie banners [10]. Cookies are small pieces of data websites use to track users and their behaviors [4]. Developers initially created cookies to help users with online shopping by allowing websites to store information on a user's machine [9]. They are now commonly used to improve website functionality and personalize user content. They usually help websites remember information about a user's visit, making it easier for a user to revisit the site. Websites usually inform visitors about cookie usage through cookie banners. These typically appear when users visit a website for the first time and contain text seeking consent from the user before deploying the cookies.

Before the establishment of the General Data Protection Regulation (GDPR), cookies were often used and abused without the user's knowledge, which led to widespread privacy concerns. When GDPR was established, websites were required to inform viewers about cookies and obtain their consent. Considering the importance of informing users of cookie usage on the website, it is essential to design cookie banners so that users can understand the implications of their decision when they choose to click "Accept" or "Agree". In a work done by Santos et al. [20], after analyzing the text of 407 banners, they found that 89% of the banners violated at least one legal requirement of the ePrivacy Directive (ePD) and the GDPR. Many of these were violations related to the purpose specificity requirement (i.e., mentioning vague purposes). These findings show how websites often employ deceptive methods to mislead users. When users encounter such practices on a website, they may unknowingly fall victim to dark patterns.

Dark patterns exploit cognitive biases [14], nudging individuals into decisions that benefit the designer or organization behind the interface, often at the expense of user autonomy and transparency. It is then imperative to clearly understand what constitutes a dark pattern. Mathur et al. [16] provide a taxonomy of dark pattern characteristics.

Asymmetric A dark pattern is asymmetric if the user interface design imposes unequal weights on the options presented. Websites prominently display choices that favor them, whereas choices that benefit users are hidden by either making them go through multiple clicks or altering the style and positioning of the option. For instance, some websites use visual techniques in their cookie banners [21] such as unequal sizes, colors, brightness, etc.

Covert A dark pattern is covert if the user interface design guides users towards specific actions or decisions without their awareness. Techniques like misdirection, disguised ads, and confusing opt-out processes are some examples of covert dark patterns [3].

Deceptive A dark pattern is deceptive if the user interface design induces false beliefs by making affirmative misstatements, providing misleading information, or withholding relevant details. Trick questions, false urgency messages, and deceptive scarcity notifications are tactics that can be categorized as deceptive.

Hides Information A dark pattern hides information if it conceals or delays the presentation of essential information to the user. This can lead to users making uninformed decisions that do not align with their best interests. Designers employ these tactics by burying crucial terms in the fine print of lengthy terms of service agreements, making it challenging for users to discover important details, or strategically placing important information in unnoticeable locations.

Restrictive A dark pattern is restrictive if only a set of choices are available to the users. Some websites strategically display the available options during checkout, pushing users towards specific add-ons or services.

Based on the characteristics mentioned above, dark patterns on a cookie banner can be based on the language clarity and the symmetry of the options available. The language or content of a cookie banner's text is said to have dark patterns if it is deceptive and hides information from the user. The options in a cookie banner are said to have dark patterns if they are asymmetric, covert, and restrictive.

1.2 Statement of the Problem

Cookies may contain sensitive user data [11], hence it becomes crucial to address the possible user consent manipulation of websites through deceptive designs in cookie banners. Although previous studies have deep-dived into how they are usually implemented and made efforts to identify them automatically, there remains an ongoing need to address this issue as digital platforms evolve. This study seeks to contribute to these ongoing efforts by improving upon Ariadne, a browser extension developed by Adorna et al. [1], which identifies deceptive patterns in the language and options in cookie banners. To address the limitations of the browser extension and extend its capabilities, this study aims to develop a web application designed to work across various browsers and devices. Moreover, the study offers a comparative analysis between the proposed tool and previous research. By evaluating and comparing the proposed tool and Ariadne, this study aims to provide insights into the effectiveness of these improvements.

1.3 Significance of the Study

The last decade has seen a significant rise in our interaction with web-based interfaces [5]. Furthermore, surveillance capitalism increasingly influences online encounters [26], which motivates various entities such as corporations and user interface designers to devise methods for gathering our data as users. Such highlights the necessity for a user-friendly means of protecting our ability to control the surveillance and data collection level that affects us.

Given the pervasive nature of digital interactions, it is important to ensure that users' personal information remains protected and that they have control over their consent choices. By addressing this essential aspect of user privacy, users can make informed decisions about their data, thereby protecting their privacy and promoting a more ethical digital environment.

2 REVIEW OF RELATED LITERATURE

This section reviews current efforts to seek solutions for automatically identifying deceptive designs in cookie banners. These initiatives aim to protect user privacy and ensure informed consent, thus promoting transparency in digital interactions. Various tools and methods have been developed to address this issue, each with strengths and limitations. These works have made significant contributions to the field, though each has its limitations in terms of accuracy and applicability.

Soe et al. [23] developed a detection tool to identify the presence of five dark patterns in cookie banners. The tool uses a machine learning pipeline, which includes feature engineering, parameter search, and a Gradient Boosted Tree classifier and evaluation. A manually labeled dataset of 300 cookie banners from news websites was used as a training set for the pipeline. The tool, however, achieved low accuracy scores, with the highest being 72%.

[23] identified three reasons the proposed detection performed poorly despite the integration of machine learning. The first reason is that it is challenging to represent cookie banners as inputs in machine-learning models. Humans perceive cookie banners and other interfaces as unified visual-language experiences. This perception is difficult to replicate in machine learning models since they typically handle cognitive tasks independently; image recognition as one task, language processing, and sentiment analysis as separate tasks. The second issue stems from the absence of standardized guidelines on how a cookie banner should look. The GDPR, for instance, does not specify elements like color choices for options, or dimensions such as height and width for cookie banners. This lack of standards complicates the development of an algorithm that can reliably extract cookie banners from websites. The third and biggest reason is the lack of a consensual definition of dark patterns among researchers. Analysis by Soe et al. [22] revealed that reviewers have difficulties agreeing upon which dark pattern is present in a cookie banner. Hence, [23] argued that improving techniques in the automatic detection of dark patterns entails refining the concepts and definitions of these patterns.

Mansur et al. [15] identified two critical challenges in automating the detection of dark patterns in user interfaces. The first challenge is how a pattern can be instantiated in many ways. This renders the process of designing an approach to detect such patterns difficult. Another challenge is the lack of a large dark pattern dataset with information mapped to a unified taxonomy of dark patterns. [15] recognize existing datasets like the one created by Mathur et al. [16]. However, the dataset lacks pertinent information, specifically the location and spatial properties of the patterns.

To address these gaps, [15] developed `AidUI`, a tool that identifies dark patterns using textual, icon, color, and spatial analysis on a user interface. This approach's fundamental idea is that multiple visual and textual cues correspond to the presence of dark patterns. For this reason, `AidUI` only requires a screenshot of the user interface as input. The study also introduced `ContextDP`, the largest dataset of fully localized dark patterns containing 501 screenshots depicting 301

dark patterns (DP) and 243 non-DP instances. Despite these innovations, the tool performed poorly, achieving an overall precision of 66%, recall of 67%, and an F1-score of 65% in detecting dark pattern instances.

Gundelach and Hermann [8] listed four methods of extracting cookie banners from a website and evaluated each of them on 1,000 randomly selected websites from the top 10,000 websites listed by Tranco. These methods are (a) Document Object Model (DOM) tree walking, which involves finding elements with the word "cookies" on a web page (b) perceptive detection, which uses image processing techniques to capture the boundaries of the consent notice (c) list-based detection, which uses CSS selectors from the manually maintained filter lists "EasyList" [18] and "I don't care about Cookies" [12] to extract the banner (d) Bidirectional Encoder Representations from Transformers (BERT)-based detection, which gets all the potential banners and uses a BERT model to determine whether that element resembles the text of a consent notice. These methods are utilized to create an automated scanning tool called `cookiescanner`. The tool extracts cookie banners from websites using the listed methods and determines if it has a decline option or use color diversion. Among the listed methods, list-based detection achieved the highest precision of 99%. Meanwhile, DOM tree walking has the highest recall of 96%. Notably, the BERT model achieves high precision for English notices but has low recall because of the low amount of candidate banners extracted.

Adorna et al. [1] developed Ariadne, a Chrome plugin that evaluates the clarity of the language used and the evenness of the options present in the cookie banner. The browser extension utilizes DOM traversal and keyword-based heuristics to extract the cookie banner from a web page. A text classifier based on a Naive-Bayes model and an image classifier utilizing convolutional neural networks (CNNs) were employed to detect the use of vague language and asymmetric options, respectively. The text classifier achieved an accuracy of 80%, precision of 60%, recall of 100%, and F1-score of 75% after training on 200 data points and testing on 10 data points. Meanwhile, the image classifier, trained on 60 images and tested on 9 images, achieved an accuracy of 56%, precision of 67%, recall of 81%, and F1-score of 73%. [1] pointed out two main reasons for the suboptimal performance: (a) the dataset used for training and testing the image classifier was inadequate to establish a reliable classifier and accurately assess its performance, and (b) the model's requirement for square images and preprocessing steps obscured the dataset, as most cookie banners are rectangular.

Ariadne addressed several shortcomings of the previous tools. However, despite its improvements, Ariadne's limitations in browser accessibility or compatibility highlight a need for a more versatile solution. Hence, a web-based tool that extends the capabilities of Ariadne by being compatible with all browsers and devices is needed.

3 VERACOOKIE: A WEB-BASED TOOL FOR DETECTING DECEPTIVE DESIGNS IN COOKIE BANNERS

Ariadne's potential is limited by it being a Chrome browser extension. Its ability to identify deceptive patterns within cookie banners, particularly on language and options present, is restricted to Chromium browsers on desktops and laptops. To address this limitation inherent in a browser-specific extension, VeraCookie was developed as a web application. VeraCookie provides the same functionality as Ariadne but is compatible with all browsers and devices. This enhanced version aims to expand the capability and accessibility of the previous plugin to alert users about deceptive patterns in cookie banners.

3.1 Architecture

VeraCookie consists of six (6) main components, each described as follows:

- A frontend that acts as the user interface.
- A controller service that validates input and extracts the cookie banner.
- An in-memory database for caching recently queried links.
- A text classifier to determine if the cookie banner text explains clearly the objectives of the data collection.
- An image classifier to evaluate the fairness of the options presented in the cookie banner.
- A classifications API that exposes the classifiers to the controller service.

All components, except for the frontend, constitute the system's backend. Figure 1 provides an overview of the system, illustrating how these components are connected.

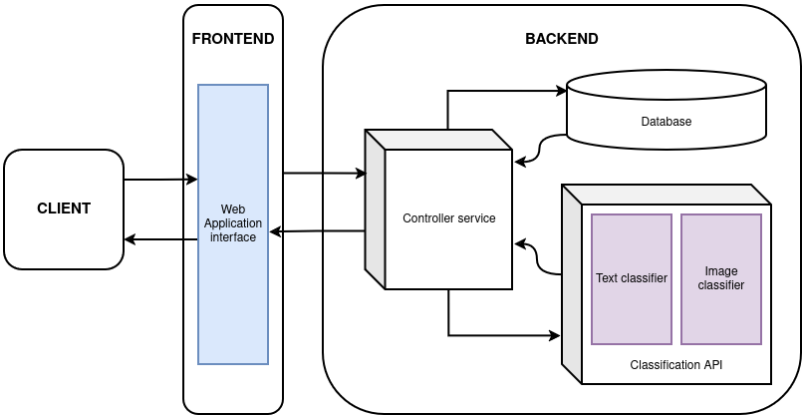

Fig. 1. VeraCookie architecture

VeraCookie architecture follows the modular architecture proposed by Adorna et al. [1]. However, there are some key differences:

- In Ariadne, the browser extension, which serves as the frontend, is responsible for extracting the cookie banner. In contrast, in VeraCookie, this extraction is handled by the controller service in the backend.
- Ariadne didn't utilize caching in their system. This is because Ariadne delegates the task of extracting the cookie banner, which uses a significant amount of resources, to the user's machine. This is in contrast to VeraCookie's approach wherein the process of extracting cookie banners is done on a single machine that hosts the controller service. Therefore, VeraCookie utilizes an in-memory database to cache recent results to offload some of the workload.

3.2 Features

The web application includes the following features:

- Extracting cookie banners from websites if present.
- Rating the clarity of the language in the extracted cookie banner.
- Assessing the symmetry of options in the extracted cookie banner.

The listed features are a subset of those provided by Ariadne. Notably, the application lacks the capability for users to manually report deceptive designs in cookie banners which is a feature available in Ariadne.

3.3 Process Flow

Figure 2 illustrates the events occurring when a user interacts with the application. Users input the uniform resource locator (URL) of the website to be checked via a form in the frontend. The controller service then receives and validates the input URL. Once validated, the controller checks if the URL has an entry in the cache. If cached, the stored result is returned to the user. Otherwise, the controller service uses headless browsing to extract any cookie banner on the site. If a banner is found, the controller service extracts its text and captures an image. The image and text are forwarded to the classifications API to identify any deceptive design elements. Two models are used: one to assess the clarity of the language in the text and another to evaluate the symmetry of the options in the image. The results, along with the image of the extracted banner, are cached and subsequently returned to the user.

4 DISCUSSION

This section provides the implementation process of VeraCookie based on its architecture presented in Figure 1. Additionally, it includes an evaluation of the implementation, examining its effectiveness, performance, and potential areas for improvement.

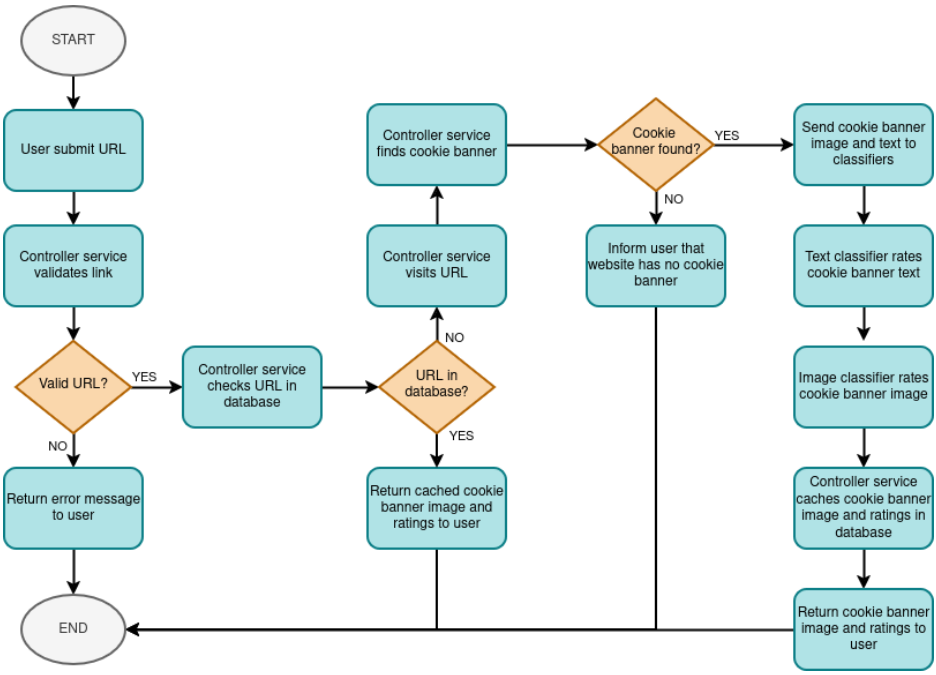

Fig. 2. Sequence of operations for VeraCookie

4.1 Frontend

The frontend is built using ReactJS[3] and styled with Tailwind CSS[4] for responsiveness. It is distributed as a Docker image to be ported and deployed easily.

As shown in Figure 3, users can enter a website URL into a form to check for deceptive patterns in the cookie banner. The URL is sent to the controller service as `FormData` where it is validated. This ensures that the URL is correctly formatted and accessible before any further actions are taken. The controller service returns the extracted cookie banner and the classification ratings to the frontend as a JavaScript Object Notation (JSON) object. These data are displayed on the frontend, as shown in Figure 4.

4.2 Controller service

As shown in Figure 1, the controller service receives and verifies input from the frontend. It then uses headless browsing to extract any cookie banner on the site. If a banner is found, the controller service extracts its text and captures an image, which is subsequently forwarded to the classifiers. In that aspect, the

[3] https://react.dev/
[4] https://tailwindcss.com/

Fig. 3. Current frontend implementation

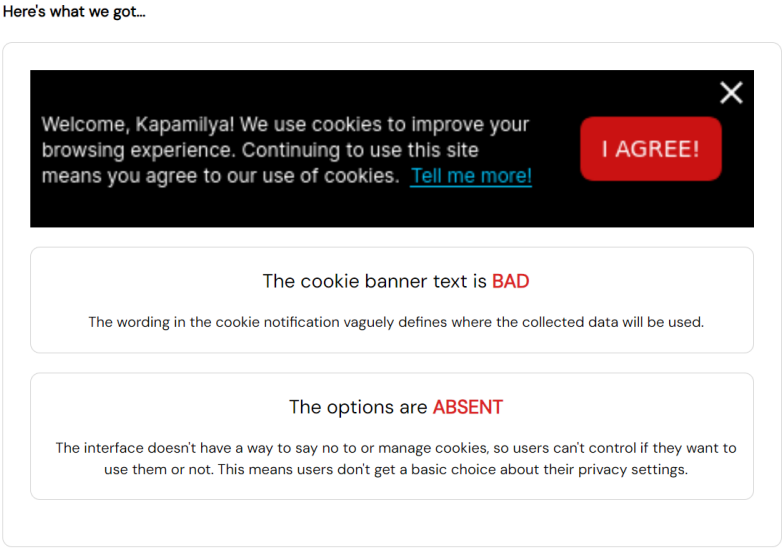

Fig. 4. Sample results for news.abs-cbn.com

controller service acts as the intermediary between the classifiers and the frontend, communicating with these components via REST API calls. The controller service exposes an endpoint to facilitate data exchange, with the specific endpoint `/analyze` utilized by the frontend to submit the user's provided link. The controller service is implemented in Spring Boot[5], a widely used Java framework, and is distributed as a Docker image for portability.

Cookie banner extraction The controller service uses Firefox and its web driver, Geckodriver[6], to perform headless browsing on user-submitted links. Selenium[7], a browser automation tool, is employed to traverse the DOM tree and extract any present cookie banner. Unlike web crawlers like BeautifulSoup[8], Selenium can render HTML documents, which is crucial since one classifier relies on the cookie banner image to make inferences. To reduce page loading time, images and videos are disabled.

The current implementation follows the cookie banner extraction steps used by Adorna et al. [1]:

1. Collect all `div` elements on the webpage.
2. Filter the collected `div`s to include only those with a `z-index` greater than 0 and containing the words "cookies", "consent", and "tracking."

Once a cookie banner is identified, its consent notice text is extracted by retrieving the `textContent` of the relevant `div`. The browser then is resized to minimize the difference between the height and width of the cookie banner, making it as square as possible. This step ensures that key features (especially buttons) remain distinguishable after resizing by the classifier (see Figure 6). This approach addresses the issue faced by Adorna et al., where resizing banners to 224×224 pixels diminished or erased important features (see Figure 5).

Fig. 5. Image pre-processing example for Ariadne

[5] https://spring.io/projects/spring-boot
[6] https://github.com/mozilla/geckodriver
[7] https://www.selenium.dev/
[8] https://pypi.org/project/beautifulsoup4/

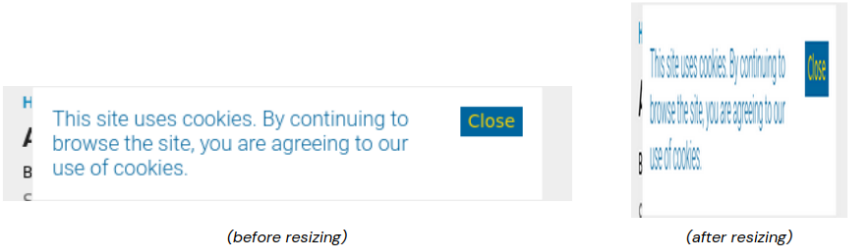

Fig. 6. Image pre-processing example for VeraCookie

Posting results to the frontend After getting the cookie banner image and text, these are sent to the models for classification via an API. The cookie banner image encoded as a `Base64` string, along with the ratings for language clarity and options weight, are returned to the frontend in a JSON object. The structure of the JSON object is detailed in Listing 1. If no cookie banner is detected, all fields in Listing 1 are set to `null`.

```
{
    image: <image in base64 string>,
    textRating: {GOOD, BAD},
    imageRating: {ABSENT, WEIGHTED, EVEN}
}
```

Listing 1. JSON object for posting results to the frontend

Caching results Performing browsing and cookie banner extraction consumes significant resources and time, even in headless mode. This is because each request initiates a new browsing instance to allow the controller service to resize the cookie banner image. To handle a potentially high number of requests efficiently, a caching functionality is implemented. Recently queried links are stored in the cache so that the application can reuse them if queried by other users. This caching is implemented using Redis[9], a popular tool for web application caching. Each cache entry has a lifetime of three days, after which the entry is deleted from the cache.

4.3 Classifications API

The controller service utilizes the classifications API to assess the clarity of language and symmetry of options in a cookie banner. Two distinct classifiers were developed for these purposes. Both are implemented in Python and are accessible via REST API calls to a Flask[10] application instance.

[9] https://redis.io/
[10] https://flask.palletsprojects.com/

The designated endpoint, `/classify`, provides access to these classifiers, allowing the controller service to relay the image and text data of the cookie banner for classification. Both the request and the response for this endpoint are in JSON format, as detailed in Listings 2 and 3, respectively. The Flask application, containing instances of the two learning models, is loaded into memory upon initiation. Furthermore, the application is packaged as a Docker image for easy portability and deployment.

```
{
    text: <cookie banner text>,
    image: <cookie banner image in base64 string>
}
```

Listing 2. JSON format of request for the classifications API

```
{
    textRating: {GOOD, BAD},
    imageRating: {ABSENT, WEIGHTED, EVEN}
}
```

Listing 3. JSON format of response from the classifications API

Text classifier The proponents utilized the criteria formulated by Adorna et al. [1] for assessing the language clarity in cookie banners. The criteria are derived from both the EU's GDPR guidelines for cookie banners and the Philippine data privacy laws, particularly Republic Act No. 10173. These are:

- Clear indication of a request for consent for non-essential cookies.
- Detailed listing of specific purposes for cookie usage and data collection.
- References to additional resources on how to access collected data, customize preferences, refuse or change preferences, and other relevant information and functions related to cookie banners.

The text classifier employs the `nltk` library to implement a Random Forest Classifier. This ensemble learning method aggregates the predictions of multiple decision trees to enhance the overall accuracy [2]. The classifier uses 100 decision trees to classify consent notice texts as either `GOOD` or `BAD`, based on their adherence to the above criteria. The default value of the `n_estimators` parameter is used for the number of decision trees, and an integer value is set for the random state to ensure consistency in training and testing sets across different runs.

The majority of the training data were scraped from various websites with cookie banners, supplemented by manually collected samples. Each data point was manually labeled as `GOOD` or `BAD` according to the specified criteria above[11]. The dataset also includes examples of GDPR-compliant and non-compliant cookie

[11] https://github.com/saytongg/veracookie/tree/main/classifiers/dataset/text_classifier/veracookie_dataset.csv

banners from Adorna et al., as well as texts from Soe et al[12]. Overall, the dataset comprised 991 cookie banner texts, divided into a $70 - 30$ train-test split, with the distribution for each set presented in Table 1.

Table 1. Data distribution in training and testing set for the text classifier

	Training Set	Testing Set
`GOOD`	399	179
`BAD`	294	119
Total	693	298

The combined dataset was imported into a Jupyter notebook[13] as a pandas `DataFrame` and standardized by removing HTML tags, non-alphabetic characters, and English stopwords. Subsequently, the texts were converted to lowercase, tokenized, and lemmatized wherein tokens are reduced to their base or root form. This ensures consistency in text data as different forms of words are treated as a single entity [17]. These tokens were then transformed into numerical vectors using the term frequency-inverse document frequency (TF-IDF) vectorization technique, which assigns numerical values to words based on their significance in a document relative to the entire corpus[19]. This approach helps in emphasizing relevant keywords over common terms. The classifier utilized the resulting numerical vectors to create a fitting model for the dataset, achieving an accuracy of 90% on the testing set. The `joblib` library was then utilized to store both the classifier and the vectorizer as it allows both components to be trained and loaded on different machines.

Image classifier The proponents adopted the options weight classes established by Adorna et al [1]. A description of these classes is given below:

- `ABSENT`, indicating the absence of an option to refuse or manage cookies.
- `WEIGHTED`, indicating that the option to accept cookies is made more obvious, more visible, or more effortless to select than the option to reject or manage it, and
- `EVEN`, indicating the consistency of options to accept and refuse cookies.

The image classifier is built upon a Vision Transformer (ViT) model, specifically the `ViT-base-patch16-224-in21k` model from Google. The model was trained on ImageNet-21k, a huge dataset that contains 14 million images and $21,843$ classes [6]. This model has 12 hidden layers and operates differently from the options weight model proposed by Adorna et al. which is based on CNN. ViTs leverage self-attention mechanisms to process image patches, treating images as sequences of tokens, while CNNs use convolutional layers to detect local

[12] https://github.com/saytongg/veracookie/tree/main/classifiers/dataset/text_classifier/ariadne_dataset.csv

[13] https://jupyter.org/

patterns and hierarchical features spatially [25]. Because of this, ViTs are better at capturing long-range dependencies, whereas CNNs excel at recognizing local features through their layered structure [13].

The pre-trained model is not designed to classify cookie banners, so it was fine-tuned using 399 manually collected and labeled cookie banner images. These images were sorted into categories: `ABSENT`, `WEIGHTED`, or `EVEN`, based on the layout and styling of options in the cookie banner. This is documented in the repository[14] at `dataset/image_classifier`.

The dataset was split into a training set ($n = 279$) and a testing set ($n = 120$) with distribution presented in Table 2. Due to the small number of training images, data augmentation was performed by vertically flipping the images in the training set. This step effectively doubled the training set size to $n = 558$. The images were afterward converted to RGB format as the model can only handle three (3) channels, then normalized to speed up the learning process, and finally resized to 224×224 pixels to achieve optimal results as the pre-trained model is trained on the same dimensions.

Table 2. Data distribution in training and testing set for the image classifier

	Training Set	Testing Set
`ABSENT`	192	41
`WEIGHTED`	192	41
`EVEN`	174	38
Total	558	120

To fine-tune the pre-trained model, the proponents followed an online guide from Winastwan [24] on using pre-trained ViTs for image classification with custom datasets. Rather than retraining the pre-trained model, a `torch` model was created using the weights and configurations from the pre-trained model. This approach enables the model to detect edges and textures, which are relevant to the task. The output layer is then augmented to produce the project-specific labels: `ABSENT`, `WEIGHTED`, and `EVEN`. To familiarize the model with this task, the Adam optimizer and softmax activation function were employed, with cross-entropy as the loss function since the task deals with multiple classes. The model was trained for 10 epochs with a small learning rate of 0.000152 and a batch size of 32, achieving an accuracy of 88% on the testing set. The model was then saved using the `save` utility by `torch` for future use, deployment, and evaluation.

4.4 Testing

The web application was evaluated in three phases: unit testing, integration testing, and acceptance testing. These are done to ensure its reliability, functionality, and alignment with user needs before being released or deployed.

[14] https://github.com/saytongg/veracookie/tree/main/classifiers

Unit Testing The confusion matrix and classification report for the text classifier are shown in Figures 7 and 8, respectively. The text classifier achieved an accuracy of 90%, precision of 89%, recall of 90%, and an F1-score of 89%. These numbers demonstrate the text classifier's effectiveness in accurately distinguishing the clarity of a cookie banner text.

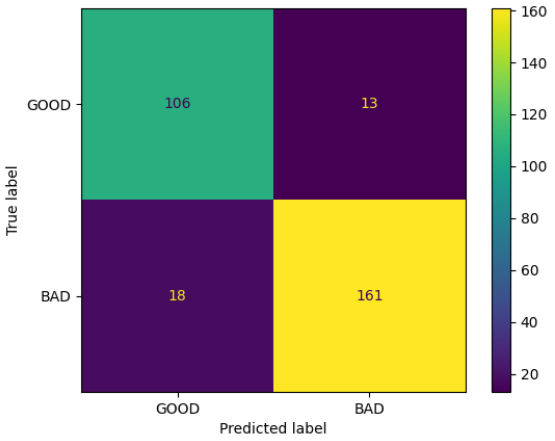

Fig. 7. Confusion matrix for text classifier

Classification Report:

	precision	recall	f1-score	support
BAD	0.85	0.89	0.87	119
GOOD	0.93	0.90	0.91	179
accuracy			0.90	298
macro avg	0.89	0.90	0.89	298
weighted avg	0.90	0.90	0.90	298

Fig. 8. Classification report for text classifier

On the other hand, the confusion matrix and classification report for the image classifier are presented in Figures 9 and 10, respectively. The text classifier achieved an accuracy of 88%, precision of 89%, recall of 88%, and an F1-score of 88%. These numbers highlight the image classifier's efficacy in accurately evaluating the fairness of options presented in a cookie banner.

The proponents also assessed the models developed by Adorna et al. using the dataset from this study. The performance of both classifiers is presented in

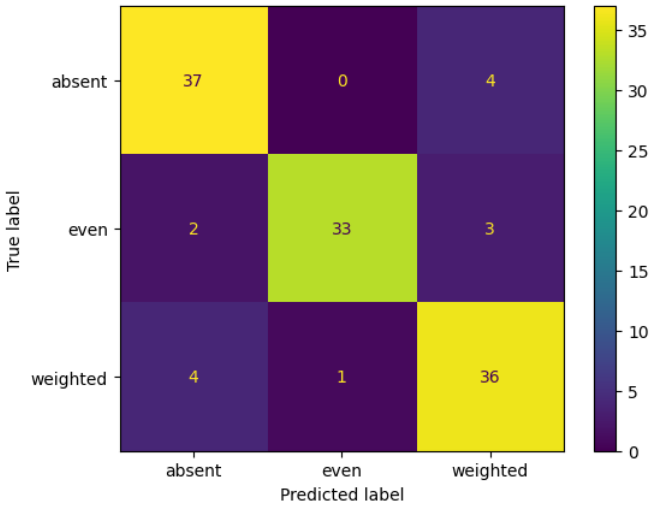

Fig. 9. Confusion matrix for image classifier

Classification Report:

	precision	recall	f1-score	support
0	0.86	0.90	0.88	41
1	0.97	0.87	0.92	38
2	0.84	0.88	0.86	41
accuracy			0.88	120
macro avg	0.89	0.88	0.88	120
weighted avg	0.89	0.88	0.88	120

Fig. 10. Classification report for image classifier

Figures 11 and 12. Their text classifier attained an accuracy of 82%, a precision of 82%, a recall of 80%, and an F1-score of 80%. Conversely, their image classifier achieved an accuracy of 68%, a precision of 78%, a recall of 67%, and an F1-score of 65%. It is important to highlight that their classifiers were evaluated using the same train-test split applied in this study.

```
Classification Report:
              precision    recall  f1-score   support

         BAD       0.84      0.69      0.76        99
        GOOD       0.80      0.91      0.85       139

    accuracy                           0.82       238
   macro avg       0.82      0.80      0.80       238
weighted avg       0.82      0.82      0.81       238
```

Fig. 11. Classification report for Ariadne's text classifier

```
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.80      0.85        41
           1       0.91      0.26      0.41        38
           2       0.54      0.95      0.69        41

    accuracy                           0.68       120
   macro avg       0.78      0.67      0.65       120
weighted avg       0.78      0.68      0.65       120
```

Fig. 12. Classification report for Ariadne's image classifier

A comparison between the classifiers is presented in Table 3. The results indicate that the classifiers developed in this study outperform those proposed by Adorna et al.. Notably, the image classifier developed in this study shows a substantial improvement in performance.

Table 3. Comparison of Performance Between Ariadne and VeraCookie

	Text Classifier		**Image Classifier**	
	Ariadne	VeraCookie	Ariadne	VeraCookie
Accuracy	82%	90%	68%	88%
Precision	82%	89%	78%	89%
Recall	80%	90%	67%	88%
F1-score	80%	89%	65%	88%

Integration Testing For the integration testing, a total of one hundred one (101) users tested the tool's function. Eight (8) news or media outlet websites based in the Philippines were selected for this test. Table 4 compares the expected and actual ratings for language clarity and options weight for each website. The expected rating categories were defined as `GOOD` or `BAD` for language clarity and `ABSENT`, `WEIGHTED`, or `EVEN` for options weight. The actual ratings were gathered from user feedback during the testing phase. For most websites, the tool's actual ratings matched the expected ratings, indicating a high level of accuracy in identifying deceptive patterns in cookie banners. This consistency was particularly evident in websites like ABS-CBN, GMA Network, TV5, Inquirer, Philippine Star, Manila Times, and Radyo Pilipinas.

Table 4. Selected Websites with their Expected Ratings and Actual Ratings

Website	**Expected Rating**		**Actual Rating**	
	Language Clarity	Options Weight	Language Clarity	Options Weight
ABS-CBN	`BAD`	`ABSENT`	`BAD`	`ABSENT`
GMA Network	`BAD`	`ABSENT`	`BAD`	`ABSENT`
TV5	`BAD`	`ABSENT`	`BAD`	`ABSENT`
Inquirer	`BAD`	`ABSENT`	`BAD`	`ABSENT`
Philippine Star	`BAD`	`ABSENT`	`BAD`	`ABSENT`
Manila Times	`BAD`	`ABSENT`	`BAD`	`ABSENT`
Radyo Pilipinas	`GOOD`	`WEIGHTED`	`GOOD`	`WEIGHTED`
Esquire Philippines	`BAD`	`WEIGHTED`	`BAD`	`EVEN`

Table 5. Accuracy Ratings Based on the Respondents' Feedback

Website	**Language Clarity**		**Options Weight**	
	Accurate	Not Accurate	Accurate	Not Accurate
ABS-CBN	89	12	88	13
GMA Network	90	11	89	12
TV5	86	15	89	12
Inquirer	88	13	82	19
Philippine Star	89	12	89	12
Manila Times	89	12	89	12
Radyo Pilipinas	97	4	98	3
Esquire Philippines	89	12	72	29

Table 5 presents a summary of the respondents' feedback on the accuracy of the tool. Radyo Pilipinas emerged as the site with the highest accuracy in both language clarity and options weight detection, while Esquire Philippines showed a notable difference in the options weight category. This suggests that the tool is highly effective in recognizing well-implemented cookie banners.

Acceptance Testing Alpha testing was conducted to evaluate user experience while using the web application. One hundred one (101) alpha testers answered a questionnaire unsupervised to evaluate the application's relevance, comprehensibility, comprehensiveness, user-friendliness, structure, speed, and layout. The alpha testers provided feedback in the form of Likert scale ratings, the averages of which are presented in Table 6.

Table 6. Average Likert scale ratings from alpha testers

Criterion	**Rating**(5)
Relevance	
The information in this website is of little use to me	2.73
This website offers information that I find useful	4.07
Comprehensibility	
The language used is easy to me	4.52
I find the information in this website easy to understand	4.53
I find many words in this website difficult to understand	2.00
Comprehensiveness	
Certain information I was looking for was missing in this website	2.73
The website provides me with sufficient information	4.01
I find the information in this website precise	4.03
User-friendliness	
I find this website easy to use	4.56
I consider this website user-friendly	4.60
Structure	
I know where to find the information I need on this website	4.29
I find the structure of this website clear	4.48
The convenient set-up of the website helps me find the information I am looking for	4.43
Speed	
I think this is a fast website	3.71
Layout	
I like the way this website looks	4.01
I find the design of the website appealing	3.86
I think this website looks unattractive	2.47

5 CONCLUSION AND RECOMMENDATIONS

The objective of this study is to improve on the previous work by Adorna et al. [1], which was a Chrome browser extension. Although the browser extension had a promising use case, its performance suffered due to the small dataset used for training and a naive preprocessing step that compromised the dataset quality. To address these issues, VeraCookie was developed. This web application incorporates new machine learning models, such as a Random Forest Classifier as the text classifier and a ViT model for the image classifier. These classifiers were

trained and tested on larger datasets, and an improved preprocessing step for cookie banner images, as detailed in section 4.2.1, was included in the controller service. Based on Table 3, these improvements were effective as the classifiers developed in this study surpassed the performance of those proposed by the previous research. Moreover, user acceptance testing yielded positive feedback indicating that the application effectively meets user needs in evaluating cookie banners for deceptive patterns. Therefore, the study successfully builds upon previous research and significantly contributes to automating the detection of deceptive designs in cookie banners.

The proponents recognize that VeraCookie still has room for improvement. To further enhance the web application, the following recommendations are provided:

- While the Random Forest Classifier showed improved performance, future researchers should explore more advanced machine learning models to further enhance the accuracy of assessing language clarity in cookie banners. Instead of the current binary classification approach, which categorizes banners as either `GOOD` or `BAD`, future work should aim to develop or find models capable of specifically identifying the types of deceptive designs present in the cookie banners. This would provide a more nuanced understanding of the deceptive elements, allowing for more targeted interventions. Advanced models, such as deep learning models or ensemble methods, could be explored to achieve this finer level of classification and improve the analysis process of the application.
- The current dataset of manually collected and labeled images may not cover the diverse designs and styles of cookie banners used across different regions and industries. Including a broader variety of datasets will help in improving the model's accuracy. Similar to the language clarity recommendation, identifying more classification types and finding or developing a capable model is one future work that should be considered.
- The application occasionally takes longer than expected to analyze websites, especially when a cookie banner is not detected. Future work could involve optimizing the current algorithms or developing new ones to improve detection rates and processing speeds. According to some remarks made by the participants of the survey, certain websites require human verification, which can either slow down the process or not do it entirely. Future researchers should explore techniques that can bypass human verification processes. Some captured images of cookie banners are obscured by ads or popups, which may lead to inaccurate data. Users may also not be able to verify the accuracy of the ratings because of this. Future developments should implement a more advanced filtering technique for this.

References

1. Juris Hannah Adorna, Aurel Jared Dantis, Rommel Feria, Ligaya Leah Figueroa, and Rowena Solamo. Developing a browser extension for the automated detection of deceptive patterns in cookie banners. In *Proceedings of the Workshop on*

Computation: Theory and Practice (WCTP 2023), pages 101–120. Atlantis Press, 2024.
2. L Breiman. Random forests. *Machine Learning*, 45:5–32, 10 2001.
3. Harry Brignull, Mark Leiser, Cristiana Santos, and Kosha Doshi. Deceptive patterns – user interfaces designed to trick you, 4 2023.
4. Aaron Cahn, Scott Alfeld, Paul Barford, and S. Muthukrishnan. An empirical study of web cookies. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 891–901, Republic and Canton of Geneva, CHE, 2016. International World Wide Web Conferences Steering Committee.
5. Pew Research Center. Internet/broadband fact sheet. https://www.pewresearch.org/internet/fact-sheet/internet-broadband/, April 2021.
6. Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
7. Sanne Elling, Leo Lentz, and Menno de Jong. Website evaluation questionnaire: Development of a research-based tool for evaluating informational websites. In Maria A. Wimmer, Jochen Scholl, and Åke Grönlund, editors, *Electronic Government*, pages 293–304, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
8. Ralf Gundelach and Dominik Herrmann. Cookiescanner: An automated tool for detecting and evaluating gdpr consent notices on websites. In *Proceedings of the 18th International Conference on Availability, Reliability and Security*, pages 1–8, 2023.
9. Amir Hormozi. Cookies and privacy. *Information Systems Security*, 13:51–59, 03 2005.
10. Privacy International. Most cookie banners are annoying and deceptive. this is not consent. https://privacyinternational.org/explainer/2975/most-cookie-banners-are-annoying-and-deceptive-not-consent, May 2019.
11. Meg Jones. Cookies: a legacy of controversy. *Internet Histories*, 4:1–18, 02 2020.
12. Daniel Kladnik. https://www.i-dont-care-about-cookies.eu/abp/, 2023.
13. Chenghao Li and Chaoning Zhang. Cnn or vit? revisiting vision transformers through the lens of convolution. *arXiv preprint arXiv:2309.05375*, 2023.
14. Maximilian Maier and Rikard Harr. Dark design patterns: An end-user perspective. *Human Technology*, 16:170–199, 08 2020.
15. SM Hasan Mansur, Sabiha Salma, Damilola Awofisayo, and Kevin Moran. Aidui: Toward automated recognition of dark patterns in user interfaces. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 1958–1970. IEEE, 2023.
16. Arunesh Mathur, Gunes Acar, Michael J. Friedman, Eli Lucherini, Jonathan Mayer, Marshini Chetty, and Arvind Narayanan. Dark patterns at scale: Findings from a crawl of 11k shopping websites. *Proc. ACM Hum.-Comput. Interact.*, 3(CSCW), nov 2019.
17. Joël Plisson, Nada Lavrac, Dunja Mladenic, et al. A rule based approach to word lemmatization. In *Proceedings of IS*, volume 3, pages 83–86. Citeseer, 2004.
18. EasyList Filter List Project. https://secure.fanboy.co.nz/fanboy-cookiemonster.txt, 2023.
19. Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer, 2003.
20. Cristiana Santos, Arianna Rossi, Lorena Sanchez Chamorro, Kerstin Bongard-Blanchy, and Ruba Abu-Salma. Cookie banners, what's the purpose? analyzing

cookie banner text through a legal lens. In *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society*, WPES '21, page 187–194, New York, NY, USA, 2021. Association for Computing Machinery.
21. Cookie Script. Are cookie banners different? https://cookie-script.com/knowledge-base/right-cookie-banner, 2023.
22. Than Htut Soe, Oda Elise Nordberg, Frode Guribye, and Marija Slavkovik. Circumvention by design-dark patterns in cookie consent for online news outlets. In *Proceedings of the 11th nordic conference on human-computer interaction: Shaping experiences, shaping society*, pages 1–12, 2020.
23. Than Htut Soe, Cristiana Teixeira Santos, and Marija Slavkovik. Automated detection of dark patterns in cookie banners: how to do it poorly and why it is hard to do it any other way. *arXiv preprint arXiv:2204.11836*, 2022.
24. Ruben Winastwan. Image classification with vision transformer. https://towardsdatascience.com/image-classification-with-vision-transformer-8bfde8e541d4, 2023.
25. Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Zhicheng Yan, Masayoshi Tomizuka, Joseph Gonzalez, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision, 2020.
26. Shoshana Zuboff. Surveillance capitalism or democracy? the death match of institutional orders and the politics of knowledge in our information civilization. *Organization Theory*, 3(3), Nov 2022.

Appendices

A Survey Questionnaire for Testing

The survey is designed to evaluate the user's experience while using the application. Users are asked to provide their feedback on these aspects using a Likert scale ranging from "Strongly disagree" to "Strongly agree." The questionnaire is based on the Website Evaluation Questionnaire [7] created by Elling et. al.

Relevance

This pertains to how useful the information provided by the application is.

1. The information in this website is of little use to me
2. This website offers information that I find useful

Comprehensibility

This pertains to how understandable the information provided by the application is.

1. The language used is easy to me
2. I find the information in this website easy to understand
3. I find many words in this website difficult to understand

Comprehensiveness

This pertains to how complete the information provided by the application is.

1. Certain information I was looking for was missing in this website
2. The website provides me with sufficient information
3. I find the information in this website precise

User-friendliness

This pertains to how easy and intuitive the application is to use.

1. I find this website easy to use
2. I consider this website user-friendly

Structure

This pertains to how clear the arrangement of the elements in the application is.

1. I know where to find the information I need on this website
2. I find the structure of this website clear
3. The convenient set-up of the website helps me find the information I am looking for.

Speed

This pertains to how responsive the application is.

1. I think this is a fast website

Layout

This pertains to how visually appealing the application is.

1. I think this website looks unattractive
2. I like the way this website looks
3. I find the design of the website appealing