Abstract—The recent privacy leakage incidences and the more strict policy regulations demand a much higher standard of compliance for companies and mobile apps. However, such obligations also impose significant challenges to app developers for complying with these regulations that contain various perspectives, activities, and roles, especially for small companies and developers who are less experienced on this matter or with limited resources. To address these hurdles, we develop an automatic tool, NL2GDPR, which can generate policies from natural language descriptions from the developer while also ensuring the app’s functionalities are compliant with General Data Protection Regulation (GDPR). NL2GDPR is developed by leveraging an information extraction tool, OIA (Open Information Annotation).

At the core, NL2GDPR is a privacy-centric information extraction model, appended with a GDPR policy finder and a policy generator. We perform a comprehensive study to grasp the challenges in extracting privacy-centric information and generating privacy policies, while exploiting optimizations for this specific task. With NL2GDPR, we achieve 92.9%, 95.2%, 98.4% accuracy in correctly identifying GDPR policies related to personal data storage, process, and share types, respectively. To the best of our knowledge, NL2GDPR is the first tool that allows a developer to automatically generate GDPR compliant policies, with only the need of entering the natural language for describing the app features. Extended version is available at [1].

I. INTRODUCTION

As the technology evolves, security and privacy have emerged as important concerns for modern systems and applications [2], [3]. Due to various data breach incidents in the past years [4], security experts and researchers have been increasingly focusing on improving the protection of user privacy. To mitigate such risks, European Union (EU) introduced General Data Protection Regulation (GDPR) on May 25, 2018, to ensure a secure and safe standard of data usage practice [5]. It imposes obligations onto organizations anywhere in the world, as long as they deal with data from the people in the EU. For example, GDPR gives individuals the right to ask for their data to be deleted and organizations are obligated to do so. The GDPR will proceed with harsh fines against those who violate its privacy and security standards [6], [7]. There are several other privacy policies (e.g., CCPA [8], CDPA [9], CPA [10]) similar to GDPR that have been developed recently.

In this study, we primarily focus on GDPR as it covers more population than the others.

Among all the challenges that the companies are facing to comply with GDPR policies, three main reasons stand out: (i) Long and complex policies. GDPR policies are very long and complex to interpret [11]. There is no simpler or less strict version for the small companies. The same GDPR policies are applied across all the companies as long as one processes data from EU citizens, which could create an excessive burden for companies with small sizes. (ii) Gap in understanding. There are often gaps in GDPR implementations [12], where organizations tend to focus on the legal aspects, contracts, security, and data protection officers, while overlooking other key elements. (iii) Lack of skill and awareness. Even worse, as developers are typically not trained against GDPR policies, researchers highlighted the major causes of poorly implemented security and privacy mechanism in apps (we refer mobile application as ‘app’) were from the problem of inexperienced, distracted, or overwhelmed developers [13].

Many popular mobile apps were found to violate GDPR policies specifically due to this, by improperly obtaining user permission on data collection or lacking the proper interaction with the user before collecting the user’s private data. For example, previously there was no option to actively opt-out from Amazon’s data collection procedures [14]. In contrast, the Waze app displays the breakdown of the purpose of using personal data, and the user must click the ‘Agree’ button to grant such permission to Waze. In this way, Waze complies with GDPR in terms of getting consent.

To this end, statistics show that around 91% companies need to recruit a dedicated team or third party firm for GDPR compliance [15]. It is costly, and many small companies cannot even afford it. Recent works tried to reduce the gap between API documentation and the corresponding implementations [16], which, however, did not offer much for complying with GDPR. For instance, they would not detect the requirement of ‘data deletion’ of the apps and provide the corresponding functionalities to comply with the data deletion policy. On the other hand, existing works on policy generation (AutoPPG [17] and PrivacyFlash Pro [18])
mainly focused on automatically generating GDPR compliant privacy policies, without checking whether the functionalities of the apps indeed comply with GDPR. Web-based privacy policy tools determine whether a particular website is GDPR compliant by checking the cookies [19], [20], asking template based questions [21]. But they are only limited to suggesting what the developer should do. Other paid tools generate the corresponding policies for the app [22].

In this work, we aim to automatically build GDPR compliant android features (definition of feature in Section III-B) from natural language descriptions to help developers alleviate the burden of excessive efforts to comply with GDPR policies. To the best of our knowledge, this is the first work on generating policies from natural language. We follow the direction of the existing template-based tools but incorporate more flexibility with natural language as the entry, leading to more diverse and customized privacy policies across different users. Specifically, we consider the following GDPR policies – retention, consent, privacy policy, access, deletion, third-party data sharing, and data processing security. We also ensure the functionality of the generated feature complies with GDPR. In this way, they won’t need to go through the long and complex GDPR policies. Using our tool, we predict the requirements of the corresponding GDPR policies, which help the developers in finding the right applicable GDPR policies. As our tool is automatically building the GDPR compliant feature, a lack of awareness of the GDPR policies would not affect the developers. We envision our tool can be the most beneficial to smaller companies or freelance developers that do not have sufficient resources for the extra effort in complying with GDPR policies. To this end, we only consider building the simple feature in this paper. Other non-GDPR-related functionalities can also be integrated with our generated features to build larger and complex features. Note that large apps would also involve more complex and specific functionalities, which will vary across different apps.

Challenges. There are several key challenges that we face while building the tool. First, since we start from natural language descriptions, unstructured description makes it difficult to detect the feature of the description, or identify the user interface (UI) element interaction on both single and multiple screens. Without accurately detecting these, it would be impossible to build an end-to-end mobile app. Second, various types of usage and processing of personal information (PII) in the app require compliance with different GDPR policies. Failure to detect the type of usage also likely leads to a violation. Third, to comply with GDPR, we need to generate a detailed description of the PII usage and privacy policies such that users of the apps can easily understand.

Our Approaches. We introduce NL2GDPR to analyze the text description of the app to generate a GDPR compliant mobile feature. Our tool has three main components: (i) information extractor, (ii) GDPR policy finder, and (iii) policy generator. First, we analyze the description to extract important information by using the information extractor component. This component is responsible for extracting UI elements, feature, and page transition. We build our information extractor with the help of a cutting-edge information extraction tool (i.e., OIA, Open Information Annotation) and a rule-based approach (Section V) with optimization and customization for our privacy-centric task. We identify all the coreference (coref) between page transitions using a recent coref tool. Second, we investigate the GDPR policy for each of the descriptions using the proposed GDPR policy finder. To do that, we employ a rule-based approach incorporating with OIA, Named Entity Recognizer (NER) tagger [23], and the same coref tool to predict the required GDPR policies. We use NER to detect the entity with which the app is sharing data. Third, based on the obtained information, we use our policy generator to create GDPR compliant privacy policies. We adapt the existing works on paraphrase generation to introduce diversity in our generated policies. We measure the quality of the generated policies (by calculating the readability scores) and select the best one to include as a privacy policy in the app. Finally, we use MIT app inventor to convert our intermediate logic into an executable apk file.

Summary of contributions:

- **Information Extractor.** We develop an end-to-end system to extract information from natural language descriptions for generating GDPR compliant features.
- **Privacy Policy Generator.** We automatically incorporate the extracted information to generate a diverse set of privacy policies. To ensure quality of the privacy policies, we also enhance the readability of the generated policies.
- **Survey.** We perform a comprehensive survey to collect natural language descriptions of 114 mobile apps.
- **Dataset.** We release the dataset and the generated data, the experimental results, app descriptions from the participants, and generated privacy policies for facilitating future advances at https://github.com/pltrees/NL2GDPR.

II. Motivating Example

We present a motivating example of a simple social app. **On the registration page, create an edittext for email and a button for creating an account. On the registration page, the user only needs to input their email address in edittext and hit the sign-up button. After that, the users will be automatically sent to the login page. On the login page, the user can provide a username in edittext and a password in another edittext. A button to push to login takes the user to the main page. The second button to sign up takes the user to the registration page.** On the Main page, create several textviews. Each textview shows news. On the Main page, the user can press the textview of news. After that, the users will be automatically directed to the news page.

From the above description, we can observe that the app has three different features - (1) Registration, (2) Login, and (3) Newsfeed. The way all the UI elements interact and connect is also described in detail. Once we extract that information, we can create the complete app flow. At the same time,
we will also have the necessary information to map those functionalities with the corresponding required GDPR policies. From the description, we can infer that the app will store and process personal information (e.g., email). So it will require the app to comply with GDPR policies (such as data deletion, access, retention, consent, privacy policies, and security of processing). With the help of the NLP tool, we can extract such key information to build GDPR compliant feature.

III. BACKGROUND

In this section, we start with introducing the terminology used in the paper, GDPR policies, and requirements, and then followed by the description of the OIA tool.

A. GDPR Policy

Alongside techniques for protecting the security and privacy of user data and systems [24]–[31], law and legal regulations are also crucial in enforcing a safe standard for practical applications. The legal requirements of GDPR policies are described in 99 different articles [5]. To illustrate the details, there are 173 recitals that provide further context and clarifications to those articles. In this work, we are highlighting the GDPR policies that are related to storing, processing, and sharing of PII information (definition of PII is described in Section III-B), as many app owners faced huge amounts of fines due to not complying with those policies previously. The investigated features in this paper are: retention, consent, privacy policy, access, deletion, sharing, security of processing.

B. Definitions of Feature and PII

We define ‘feature’ as the main functionality of a description. We take the following app description as an example: “I want to create a ‘Registration’ page for my app. The first page of ‘Registration’ will have two buttons and three edittext. The edittext is used for entering the user’s name, email, and password, respectively. Click the button of ‘sign in’ to jump to the ‘login’ page. When you click the button of ‘sign up’, you can register your account.” This illustrates the process of registration in the application. Here, we consider ‘registration’ as the feature of this description.

According to GDPR, any information that can be used to identify an individual is considered as PII [32]. We list the following information as PII in this paper – username, firstname, lastname, name, email, mail, address, country, state, zipcode, city, county, age, location, birthdate, ipaddress [33], [34]. Whenever an app wants to collect these information, it needs to comply with GDPR. In other words, storing, processing, and sharing these information requires extra attention. Our tool detects the presence of these PII and stores/processes/shares the data in a GDPR compliant manner in the generated app.

C. Measuring Readability

Readability defines how easy it is to understand one’s writing. The higher readability of a sentence, the easier people will be able to understand. Some of the most popular formulas to calculate the readability include Dale-Chall Score [35], Gunning’s FOG Index [36], and Flesch Reading Ease [37], which are utilized to assess the readability level of our generated policies in this paper. We expect the generated policies to be understandable by a seventh or eighth grade student. Thus, we set the readability threshold values accordingly to reflect this readability level.

D. OIA Tool

Open Information Annotation (OIA) [38], [39] is the recently proposed approach for building OIE systems. It represents all the information in a sentence into a predicate/function-argument and expresses them in a graph. The OIA graph is generated with a cutting-edge neural predictor. As shown in Figure 1, the input sentence is encoded with BERT [40]. As a result, each word is represented with a word embedding. Then three neural components predict the topology, edge label, and node label of the graph, respectively.

![Fig. 1: The process of the neural predictor converts a sentence to an OIA graph.](image)

Among these, the node label predictor is a neural multi-class classifier that generates a tag for each word node based on the word embeddings. For instance, ‘noun’ nodes (ellipse in Figure 1) represent for entities. The ‘event’ nodes (rectangle in Figure 1) and their arguments represent interactions. The ‘prepositional’ nodes connect the event and its attributes (e.g., time, location, manner, etc.). The topology predictor is a
matrix predicted based on the word embeddings. The element at row $i$ and column $j$ is used to predict whether there is a directed edge between word $i$ and word $j$.

The edge label predictor is a tensor predicted based on the word embeddings and used to select labels for edges between word pairs. A typical edge label like ‘pred.arg.x’ shows the tail node is the $x$th arguments of the head node. For the event node, the first argument and is the subject, and the second argument is the object. For the prepositional nodes, the first argument is the event, and the second argument is the attribute of the event.

The results of these predictors are merged together as the directed graph with word-level nodes. Then the tool needs to merge the word-level graph (upper left in Figure 1) into a phrase-level graph whose nodes contain descriptions of components or interaction with more than one words (upper right in Figure 1). In the word-level graph, there are two types of edges, i.e., intra-phrase and inter-phrase edges. Intra-phrase edges (edges with label ‘next_word’) are the basis of graph structure changing. Word-level nodes connected by ‘next_word’ are merged into phrase-level nodes. In contrast, inter-phrase edges (edges with other labels) remain unchanged. Finally, the tool can get the OIA graphs that describe the interactions or system actions.

The OIA graph can be used to efficiently and effectively harvest ‘event’ nodes and their argument sub-trees as the information extraction result. Evaluating on the recently widely-used open information extraction benchmark Re-OIE2016 [41], OIA achieves the best result and improves accuracy by about 2% from the prior best result reported in [42].

IV. DATA COLLECTION

We collect natural language descriptions of 114 apps by performing a survey. Each of the participants is required to have at least two years of development and extensive user experience on mobile applications. Such expertise ensures that the participants have the necessary knowledge of the mobile applications and processing of different user data for this survey. We perform the survey in two phases. In the first phase, we ask five participants to participate. We collect 66 app descriptions which contain 258 sentences (containing 23 features) from them. In the second phase, we collect the data of 48 apps (contain 396 sentences) from different participants, while ensuring each app is described by at least three participants. In the study, we ask the participants to describe each app in natural language. We provide them a guideline with a list of information that they need to use in their description. Our information contains (i) feature list, (ii) UI elements, and (iii) PII data. To help them write high-quality descriptions, we have included a few examples of different apps along with the corresponding screenshots. It will also help them to understand the guideline for writing the description correctly. It took only 15-17 mins on average to describe each app.

V. SYSTEM DESIGN AND IMPLEMENTATION

The system architecture of NL2GDPR is shown in Figure 2. Our system consists of three key components: (i) Information Extractor, (ii) GDPR Policy Finder, and (iii) Policy Generator. As described in Section IV, we select 66 apps from phase 1 to build the component’s logic of our tool. Specifically, we devise the rules for each component’s detection mechanisms by analyzing these 66 app descriptions as the training data. In the following, we discuss the detailed implementations of these components.

A. Information Extractor

To build an app, we need three types of information: page transition, UI element, and feature. We collect these information using three different sub-components of the information extractor component: Page Information Extractor, UI Information Extractor, and Feature Information Extractor. The complete process is described in Figure 3.

Existing parsers (e.g., dependency parser) require craft rules based on their results. That means we have to read the descriptions and summarize expression habits to bridge the parsing results and the information of interest. Specifically, (i) to recognize feature names, we need to merge all kinds of nominal syntactic tags with their decorative sub-graph by rules. Alternatively, we can train a recognition model with enough labeled data. (ii) To recognize the description of user interactions, we need to merge verbal phrases and analyze the semantic roles of verbal words to avoid decorative verbs (e.g., ‘login’ in ‘login button’). (iii) Finally, we must coordinate the results of (i) and (ii) to eliminate conflicts and ensure a corresponding relationship between user interactions and app components. These processes will involve data annotation, rule crafting, and analysis of part-of-description, syntax, and semantics. In our task, we have to carefully design the implementations and adapt tools with customization for these steps, as any step might bring errors and hurt the performance of the entire process. In the following, we discuss the details of the three sub-components.
One problem in OIA is that the mentions extracted facts may be mentions of pronouns like “it”, “that”, “they”, which do not involve the information about the referred real entity like “the registration page”, “the home page”. To address this issue, we adapt the coref tool described in [43] to resolve the pronouns to the real entity.

For every noun node, we check the coref information. In the ‘findCorefInfo()’ algorithm, we traverse the full sentence to detect whether the corresponding noun node has any coref page name. If it is present, we assign that information as page.Name. Otherwise, we extract the page name by using our rule-based approach. In this approach, we search for the node which contains ‘page’ word to determine page name. For the sentences with transition functionality available, the number of extracted pages became more than two. In those cases, we extract the first page name information as the ‘current’ page and all the later information as the ‘transition’ page information. Since we can expect the current page name to be mentioned before the transition page by a developer, which is also consistent with our observations of our collected descriptions, it performs well by selecting the first page information as current page information.

2) UI Information Extractor: Each description includes multiple sentences for describing the detailed interaction of the UI elements. From each sentence, we search the presence of UI elements by using a keyword-based approach. While performing the survey, we presented the participants with a list of UI elements. By analyzing the training data (Section V), we observe that we can use the keyword list to collect all the UI elements in a sentence.

In the survey, we show a list of UI elements that the participant is allowed to use while describing the app. Our full list of UI elements are: [‘TextView’, ‘EditText’, ‘Button’, ‘ImageButton’, ‘RadioButton’, ‘RadioGroup’, ‘CheckBox’, ‘AutoCompleteTextView’, ‘ProgressBar’, ‘Spinner’, ‘TimePicker’, ‘DatePicker’, ‘SeekBar’, ‘AlertDialog’, ‘Switch’, ‘switchbutton’, ‘RatingBar’, ‘Map’, ‘RadioButton-CONTROL’]. From our investigation, we find the participant use all the UI elements from this list. We thus select this list to find the presence of UI elements by investigating the exact match.

Once we find an exact match with the listed keyword, we detect that as a UI element. Using the page detector component, we determine the name of the page where the UI elements are located.

3) Feature Information Extractor: We predict the feature information in three steps. First, using the OIA graph, we detect the ‘noun’ node in the sentences. Second, we use our rule-based approach to find the presence of the listed keywords in the corresponding sentences. When we find an exact match of the keywords, we mark the sentence exhibiting the corresponding feature. Third, we count the frequency of the matched feature inside a description. Since not all the sentences indicate the feature information and the feature information may spread across multiple sentences, we aggregate all the information extracted to infer the feature of an app description. Our intuition is that the name of the final feature will appear the most frequently inside a description. Therefore, we consider the feature that is predicted the most in the sentences as the final feature of the description. For example, if the keyword of ‘registration’ shows up in the sentences of a description twice while the keyword of ‘login’ appears only once, we mark that description as ‘registration’. 

![Diagram of the OIA process](image-url)
TABLE I: Rule-based approach for finding GDPR policy. S = Storage, P = Process, T = Third-party sharing.

| Type | Events            | Features                                                                 | Rules                                             |
|------|-------------------|--------------------------------------------------------------------------|---------------------------------------------------|
| S    | Store, Save, Upload, Input, Register, Create, Record | 'registration', 'user profile', 'status updates', 'comments', 'address book', 'change password', 'user status', 'blog writing', 'add new friends', 'notes', 'create new post' | <feature>: <event> <PII> |
| P    | Show, View, Display, Exhibit | 'news feed', 'home feed', 'summary of the day', 'comments', 'address book', 'login', 'people nearby', 'emoticon input', 'product scan', 'chat with friends', 'user friends list', 'search for people nearby', 'app purchase', 'news recommendation', 'share', 'review' | <feature>: <event> <PII> |
| T    | Share, Send       | 'third-party integrations', 'app purchase', 'share', advertisement        | <feature>: <event> <PII> <party>                  |

B. GDPR Policy Finder

After extracting all the features and page-related information (using the proposed information extractor), we investigate the required GDPR policies for each of the descriptions.

We divide this process into three parts: (i) identify PII information, (ii) identify usage of data, and (iii) generate intermediate representation of the required GDPR policy. The complete process is shown in Figure 4. We describe each of them in the following.

![Fig. 4: Using OIA to detect the required GDPR policy from description.](image)

1) Identify PII information: GDPR policies are applicable when the app uses PII (Section III-B). Thus, we need to identify whether the given description contains any PII or not. First, we extract all the noun nodes using OIA. From our analysis, we observe that PII always locates in the noun nodes. Then, we use our keyword-based approach to find the exact match of the PII keywords. We build a PII mapping keyword set to find the PII information in the description.

2) Identify Usage of Data: Once we detect PII, then we start to investigate how that information is being used. From our investigation on the training sample, we find that there could be three different types of usage – storage, process, and third-party sharing. Sometimes, the app will store the user information (e.g., user credentials during registration). We define that as ‘data storage’. For completing user requests, the app may need to use user information (e.g., matching username and password during login). We define such usage as ‘data process’. For improving user experience (e.g., sharing a user email address with the product manufacturer for advertisement purposes), the app may share user data with third parties. We denote such sharing as ‘third-party data sharing’. Let’s consider the following two examples for better illustration.

**E1** ‘login’ page, user can provide her username in edittext and password in another edittext. Clicking the ‘login’ button will take the user to the ‘home’ page.

**E2** ‘register’ page, user provides username in the first edittext. Second edittext user provides ‘first name’, third edittext user provides ‘last name’, fourth edittext user provides ‘password’, fifth edittext user confirms ‘password’ again. Button at the bottom which is ‘register’ will store all the information to the server.

In the first example, the app does not store any personal information. Whereas in the second example, the app stores personal information (such as first name, last name, password, etc.). Here, the app needs to comply with GDPR policies (according to P1-P5, P7).

We illustrate the rule-based approach in Table I where we use it for mapping event types and feature information with the type of data usage. We incorporate that information and look for which event keywords (listed in Table I) are presented in the corresponding sentence. From analyzing the sentences, we find the ‘store’ keyword, which indicates ‘storage’ type data usage. By following the same approach, we compute the type of data usage in each sentence. Ultimately, we compute all the unique data usage types to determine the final set of data usage types, which determines the required set of GDPR policies.

3) Generating Intermediate Representation: Once we find all the information from the other two sub-components, we mapped these to the corresponding GDPR policies based on Table II. The table shows how each of the GDPR policies is linked to the corresponding data usage type. For example, according to P2, the user should grant permission before the app can use the PII. To make the app compliant with P2, we introduce ‘consent’ functionality whenever the app needs to take input from the user-end. This policy is also applicable where the app shares PII with other third parties. So in our generated app, we implement the consent also whenever the app is sharing data with other third parties. By following the same direction, we develop the functionalities in Table II for each of the GDPR policies as an intermediate representation. This ensures the generated app’s functionality complies with GDPR policies.
TABLE II: Mapping of usage of data with the GDPR policies.

| Type | Policy Functionality |
|------|----------------------|
| S    | \( period > retention \& \& exec(delete(data)) \) |
| S, T | \( data_{receive} \& \& \& (collect(data) || share(data, third-party)) \) |
| S, P, T | \( readability(policy_{privacy}) \& \& contain(PII_{collected, policy_{privacy}}) \& \& contain(PII_{stored, policy_{privacy}}) \) |
| S    | \( req_access(data) \& \& exec(req_access(data)) \) |
| S    | \( req_delete(data) \& \& exec(req_delete(data)) \) |
| P    | \( send(encrypt(data), third-party) \) |
| S, P | \( send(encrypt(data), server) \) |

TABLE IV: Performance evaluation in extracting UI elements.

| Type          | #Matched | #Not Matched | Accuracy |
|---------------|----------|--------------|----------|
| Current       | 357      | 59           | 90.2%    |
| Transition    | 347      | 49           | 87.6%    |

C. Policy Generator

We collect the template of privacy policies from the GDPR website online (https://gdpr.eu/privacy-notice/). After analyzing the template, we identify information that needs to be consistent with the app functionalities. In the privacy policy, an app is expected to state the reason for collecting each PII clearly, which we refer to as ‘purpose’. We generate the purpose with the help of an existing paraphrase generator tool [44] and seed purposes. We collect different policies (from popular apps) for different features, which we use as seed data for the paraphrase generator tool. We create the purposes for all the different features considered in this work. We collect those from the privacy policies of ten most popular mobile apps in the google play store, i.e., Facebook, Instagram, WhatsApp, Google, TikTok, Snapchat, Pinterest, Twitter, Reddit, Skype. For each sentence inside these policies, we generate ten different paraphrases. Then, we investigate whether the generated purpose preserves the actual meaning or not. Due to the difference in the wordings, the rephrased sentence’s meaning may change slightly, resulting in ambiguity. But in the privacy policy, we have to ensure that the readers can clearly understand the purpose of collecting PII. Thus, we measure the quality of the paraphrase sentences. After generating an OIA graph from the original sentence, we identify the representative noun nodes. Note that a node is defined as a representative noun node when it contains more than two edges (either incoming or outgoing or both) in the OIA graph or PII information. From our investigation, we find these noun nodes are essential in preserving the sentences’ actual meaning. If the representative noun nodes are missing in the generated paraphrase graph, we mark that as ‘not preserving’. To ensure the readability of the generated purpose, we check the readability level of each of the sentences. We calculate three scores for each generated purpose, i.e., Dale-Chall Score [35], Gunning’s FOG Index [36], and Flesch Reading Ease [37], as described in Section III-C. We examine whether the generated purpose is easily understandable by at least a seventh or eighth grade student by matching the corresponding value for each of these scores. Consequently, we create a diverse set of readable GDPR compliant policies.

However, if our tool fails to predict no UI elements or features or page information in the description, it will automatically ask the user to revise the description and provide the refined description by highlighting the specific sentence for revision and providing the reasons (e.g., missing UI element, missing feature, missing page information, vague UI element, vague feature, vague page information). This functionality is easily implemented by using the OIA parsing results. In addition, we plan to create a portal where users can request new features or UI elements for facilitating future development.

Once we extract all the information using NL2GDPR, we convert all the functionalities to apk file (which can be installed in Android phones) using MIT app inventor [45], a popular tool among app developers. We use this tool to generate an executable apk file for mobile devices running Android OS.

VI. EVALUATION

We present the detailed evaluations of NL2GDPR in generating a GDPR compliant app. The tool suite is implemented as a Python (3.7) prototype on a Desktop PC with 16 GB of RAM and a 3.1 GHz Intel Core i5 processor, running Ubuntu 18.04 LTS. We use all the data from the first phase as the ‘training data’ (66 app), and the second phase as the ‘test data’ (36 app). None of the participants involve in both phases.

A. Evaluation of Information Extractor

Our information extractor component consists of three subcomponents. Now, we evaluate the performance of those.

1) Page Information Extraction: In the descriptions of 48 apps, we had the information of 396 total pages available. The performance of page information extraction is presented in Table III. It can be seen that by using our tool, we were able to detect 357 current page and 347 transition page information successfully, while missed the current page information 39 times and transition page information 49 times, respectively. In summary, NL2GDPR achieves 88.9% accuracy (90.2% for current and 87.6% for transition) in identifying page information.

TABLE III: Evaluation of extracting page information.

| Type          | #Matched | #Not Matched | Accuracy |
|---------------|----------|--------------|----------|
| Current       | 357      | 39           | 90.2%    |
| Transition    | 347      | 49           | 87.6%    |

2) UI Information Extraction: In our test set, we have 48 app descriptions that contain interactions with 396 sentences. These descriptions contain 497 UI elements (from 396 sentences) in total. In Table IV, we illustrate the detailed performance of extracting UI elements for our tool. Our tool is able to identify 476 (from 388 sentences) UI elements correctly. It fails to detect 21 UI elements from 8 different sentences. By investigating these misclassifications, we find that

TABLE IV: Performance evaluation in extracting UI elements.

| #Matched | #Not Matched | Accuracy |
|----------|--------------|----------|
| 476      | 21           | 95.8%    |
NL2GDPR, in rare cases, fails to differentiate the descriptions for the UI elements’ position from the entity of UI elements. For example, “In this page, there is an ‘input’ edittext, and alongside the edittext, there is a ‘search’ imagebutton.” In the second half of the sentence, the participant attempts to describe the position of the imagebutton corresponding to the edittext, where the edittext refers to the one in the first half as opposed to a new edittext. This confuses our tool, such that it creates a new ‘edittext’ element in the UI. But these are some rare cases as NL2GDPR is able to achieve 95.8% accuracy.

3) Feature Information Extraction: In this section, we evaluate the performance of our tool from the perspective of extracting feature information. We extract the key information related to each feature from each sentence of the descriptions. So we evaluate the performance of our tool in terms of detecting features based on each description.

**TABLE V: Performance evaluation in identifying feature.**

| #Feature | #Matched | #Not Matched | Accuracy |
|----------|----------|--------------|----------|
| 48       | 44       | 4            | 91.1%    |

From Table V, we can see that NL2GDPR is able to identify the majority of features correctly with a 92% accuracy. For example, “In my ‘login’ page, it has 2 edittexts and 4 buttons. The edittexts are used to enter ‘email’ and ‘password’. There is a ‘button’ of ‘login’ in the upper right corner. Once the user clicks the button of ‘login’, it will send the user’s ‘email’ and ‘password’ to the server. There are 2 buttons below the edittext to avoid forgetting the account or password. Press the button of ‘Forgot Password’ to jump to the ‘Forgot Password’ page, and press the button of ‘Use Device Code’ to jump to the ‘Use Device Code’ page.” After analyzing all the sentences in this description, our tool predicts the feature as ‘login’.

**B. Evaluation of GDPR Policy Finder**

We evaluate NL2GDPR regarding the detection of GDPR policies, as shown in Table VI. Our GDPR policy finder can successfully identify whether a single app description exhibits the characteristics of storage, process, third-party data sharing, or nothing.

**TABLE VI: Performance evaluation of GDPR Policy Finder.**

| Type              | #Matched | #Not Matched | Accuracy |
|-------------------|----------|--------------|----------|
| Storage           | 368      | 28           | 92.9%    |
| Process           | 377      | 19           | 95.2%    |
| Third-party Sharing | 390     | 6            | 98.4%    |

From Table VI, we can observe that NL2GDPR achieves about 93%, 95%, and 98% accuracy for detecting storage, process, and third-party sharing, respectively. Overall, our tool correctly detects the 368 storage, 377 processes, and 390 third-party sharing data usages.

**C. Evaluation of Policy Generator**

With our paraphrase generator, we generate 300 different purposes for 23 features. After evaluating them, we filter out 136 generated purposes that do not convey the original information of the corresponding purposes. That indicates either they do not have the representative noun nodes or their readability is less than the threshold value. We evaluate the remaining 164 generated purposes manually and find our tool is successful in generating 149 purposes. Thus, NL2GDPR achieves 90.9% accuracy in generating the purposes of the privacy policies.

**D. End-to-end Validation of NL2GDPR**

We select 48 app descriptions and perform an end-to-end evaluation of our tool. We manually annotate those apps to the corresponding required policy. For example, if a feature shares user’s ‘email’ with third parties, we mark P2, P3, P6, and P7 as the required policies that the generated feature should comply with. Then, we evaluate each generated feature whether it contains these policies or not.

**TABLE VII: End-to-end evaluation of NL2GDPR.**

| Policy | #Matched | #Not Matched | Accuracy |
|--------|----------|--------------|----------|
| P1     | 43       | 5            | 89.6%    |
| P2     | 43       | 5            | 89.6%    |
| P3     | 42       | 6            | 87.5%    |
| P4     | 43       | 5            | 89.6%    |
| P5     | 42       | 6            | 87.5%    |
| P6     | 44       | 4            | 91.7%    |
| P7     | 44       | 4            | 91.7%    |

In Table VII, we show that the number of listed policies (P1-P7) are correctly identified by using our tool. We can observe from this table that our tool can detect all the seven policies with an average of 90% accuracy. Since the tool can automatically ask the developer to revise any missing information as described in Section V-C, if we assume all the descriptions are revised according to the requirement and no feature or UI element is outside the supported lists of our tool, we expect NL2GDPR can achieve much higher accuracy in predicting all the seven policies. With the help of NL2GDPR, we are able to mitigate the hurdle of GDPR compliance. Developers don’t need to go through long and complex GDPR policies. It is the responsibility of the NL2GDPR to ensure the generated feature complies with GDPR regulations will save the developers time in understanding the proper way of protecting user personal information. The developers of NL2GDPR have gone through all the GDPR policies and have participated in several discussions related to GDPR to help implement the corresponding requirement of the GDPR policies for a particular feature.

**VII. DISCUSSION**

We want to address a few limitations of our tool. First, a rule-based approach generally suffers from a lack of coverage, which might lead to false negatives due to insufficient rules. However, from Table VII, we can observe that our tool achieves a decent performance with accuracy lies between 88%-92% for seven different policies. Second, we are only focusing on building GDPR compliant features, while all the
server-side computation is beyond the scope of this paper. For example, whether the developer actually deletes the personal data from the server is out of the scope of this paper.

VIII. CONCLUSION

We proposed NL2GDPR, the first automatic policy generation tool for an app to create GDPR compliant policies from natural language descriptions. It is designed to assist app developers in ensuring compliance with GDPR for the generated app, which will significantly alleviate the burdens for the developers. To build NL2GDPR, we adapted an information extraction tool, OIA, and developed several optimization techniques for our privacy-centric task [38], [39]. By using the extracted information from the natural language descriptions, NL2GDPR finds the associated GDPR policy and generates compliant privacy policies. The effectiveness and advantages of NL2GDPR are comprehensively evaluated, which shows that NL2GDPR can achieve superior performance in detecting various GDPR policies. Our results confirm NL2GDPR can successfully generate GDPR compliant policies and functionalities for the app. The full paper is available at [1].

REFERENCES

[1] F. H. Shezan, Y. Lao, M. Peng, X. Wang, M. Sun, and P. Li, “NL2GDPR: Automatically Develop GDPR Compliant Android Application Features from Natural Language,” arXiv preprint arXiv:2209.13361, 2022.

[2] K. Doan, Y. Lao, and P. Li, “Backdoor attack with imperceptible input and latent modification,” in Advances in Neural Information Processing Systems (NeurIPS), 2021, pp. 18944–18957.

[3] K. Doan, Y. Lao, W. Zhao, and P. Li, “LIRA: learnable, imperceptible and robust backdoor attacks,” in Proceedings of ICVV, 2021.

[4] “Cambridge analytica and facebook: The scandal and the fallout so far,” https://www.nytimes.com/2018/04/04/us/politics/cambridge-analytica-scam-fallout.html, April 4 2018.

[5] “General Data Protection Regulation,” https://gdpr-info.eu/, 2021.

[6] Kavya, “15 major GDPR fines of 2020 & 2021,” https://www.cookielaw.io/15-major-gdpr-fines-2020-2021/, May 3 2021.

[7] “Austrian DPA has option fine google €6 billion,” https://tnaoy.by/en/austrian-dpa-has-option-fine-google-e6-billion, May 6 2021.

[8] “California consumer privacy act,” https://oag.ca.gov/privacy/ccpa, 2021.

[9] “Virginia’s new consumer data protection act,” https://www.natlawreview.com/article/virginia-s-new-consumer-data-protection-act, 2021.

[10] “Colorado privacy act becomes law,” https://iapp.org/news/a/colorado-privacy-act-becomes-law/, 2021.

[11] D. S. Guaman, J. m. del Alamo, and J. C. Caiza, “GDPR compliance assessment for cross-border personal data transfers in android apps,” IEEE Access, vol. 9, pp. 15 961–15 982, 2021.

[12] S. Kununka, N. Mehandjee, and P. Sampato, “A comparative study of android and ios mobile applications’ data handling practices versus compliance to policy,” Springer, 2017.

[13] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek, and C. Stransky, “You get where you’re looking for: The impact of information sources on code security,” in Proceedings of the IEEE Symposium on Security and Privacy (SP), 2016, pp. 289–305.

[14] “Apps violating GDPR regulations,” https://www.privacypolicies.com/blog/gdpr-compliance-appl, January 22 2021.

[15] DrFazal, “The cost of GDPR compliance,” https://drfazal.medium.com/the-cost-of-gdpr-compliance-5e58c2b5232e, Jan 31 2020.

[16] D. C. Nguyen, D. Wermke, Y. Acar, M. Backes, C. Weir, and S. Fahl, “A stitch in time: Supporting android developers in writingsecure code,” in Proceedings of CCS, 2017, pp. 1065–1077.

[17] L. Yu, T. Zhang, X. Luo, and L. Xue, “Autoppy: Towards automatic generation of privacy policy for android applications,” in Proceedings of the 5th Annual ACM CCS SPSSM, 2015, pp. 39–50.