Consistency Analysis of Data-Usage Purposes in Mobile Apps

Duc Bui, Yuan Yao, Kang G. Shin
University of Michigan
Ann Arbor, Michigan, USA
{ducbui,kelyao,kgshin}@umich.edu

Jong-Min Choi
Samsung Research
Seoul, Republic of Korea
jminl.choi@samsung.com

Junbum Shin*
CryptoLab
Seoul, Republic of Korea
junbum.shin@cryptolab.co.kr

ABSTRACT

While privacy laws and regulations require apps and services to disclose the purposes of their data collection to the users (i.e., *why do they collect my data?*), the data usage in an app's actual behavior does not always comply with the purposes stated in its privacy policy. Automated techniques have been proposed to analyze apps' privacy policies and their execution behavior, but they often overlooked the *purposes* of the apps' data collection, use and sharing. To mitigate this oversight, we propose PurPliance, an automated system that detects the inconsistencies between the data-usage purposes stated in a natural language privacy policy and those of the actual execution behavior of an Android app. PurPliance analyzes the predicate-argument structure of policy sentences and classifies the extracted purpose clauses into a taxonomy of data purposes. Purposes of actual data usage are inferred from network data traffic. We propose a formal model to represent and verify the data usage purposes in the extracted privacy statements and data flows to detect *policy contradictions* in a privacy policy and *flow-to-policy inconsistencies* between network data flows and privacy statements. Our evaluation results of end-to-end contradiction detection have shown PurPliance to improve detection precision from 19% to 95% and recall from 10% to 50% compared to a state-of-the-art method. Our analysis of 23.1k Android apps has also shown PurPliance to detect contradictions in 18.14% of privacy policies and flow-to-policy inconsistencies in 69.66% of apps, indicating the prevalence of inconsistencies of data practices in mobile apps.

CCS CONCEPTS

• **Security and privacy → Privacy protections**.

KEYWORDS

Data-usage purposes; privacy policies; consistency analysis; data flow; mobile apps

ACM Reference Format:
Duc Bui, Yuan Yao, Kang G. Shin, Jong-Min Choi, and Junbum Shin. 2021. Consistency Analysis of Data-Usage Purposes in Mobile Apps. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3460120.3484536

*This work was done while the author was affiliated with Samsung Research prior to joining CryptoLab.

CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8454-4/21/11…$15.00
https://doi.org/10.1145/3460120.3484536

1 INTRODUCTION

The Federal Trade Commission (FTC) has relied on privacy policies written in natural language as a primary means to check and inform users how and why apps collect, use and share user data [16]. Since purposes of data collection and use/sharing are key factors for users to decide whether to disclose their personal information or not [42], it is important for apps to make the users aware of, and consent to them. For example, users would more likely to agree to provide their location for receiving an app's services rather than for advertising purposes. Moreover, while the purposes of data collection, use and/or sharing are specified in the apps' privacy policies, the apps' actual execution behavior may deviate from their specifications in the policies.

Despite its importance, little has been done on checking the consistency between the *purposes* stated in the privacy policies and the actual execution behavior of apps. Prior studies [8, 7, 65, 70, 72, 75] overlooked the *purposes* and *entities* whose purposes were served. Furthermore, the assumption that data sent to an entity is always used for *any* of the receiver's purposes may not hold when the external service processes the data for the app's purposes. For example, the data sent to an analytic service should be used for the app to analyze its usage trend, not for the analytic service's purposes such as delivering personalized advertisements.

A key question is then: *Can we automatically check whether the purposes of actual data usage comply with those stated in privacy policies or not?* The first challenge in answering this question is to achieve a clear interpretation of the privacy policy and detect contradictory privacy statements which, if exist, will make the disclosure of data flows ambiguous. The second challenge is to extract the purposes of the actual data flows from the app behavior and compare them with (potentially contradictory) privacy statements.

Analyzing fine-grained purposes of data usage yields a fundamentally different and more complete interpretation of privacy policies than purpose-agnostic approaches, such as PolicyLint [7] and PoliCheck [8]. Let us consider the following policy statement from a popular app on Play Store with more than 1M installations.

Example 1: "We do not share personal information with third parties for their own direct marketing purposes."

PurPliance interprets this example as third parties may collect personal data but do not use it to deliver their own advertising, which is part of marketing purposes. Therefore, PurPliance flags a contradictory data-usage purpose in another statement stating that the app "may share your personal data with third-party advertising partners to serve personalized, relevant ads." Purpose-agnostic approaches [8, 7] narrowly interpret Example 1 as the app would not share any personal data. Such approaches do not accurately detect

the contradiction of the advertising usage purpose and generate lots of false positives because the example would then contradict any other statements about sharing of the user's personal data.

We present PurPliance, an end-to-end fully automated system that detects contradictory privacy statements and inconsistent app behaviors. In the system workflow (depicted in Fig. 1), contradiction/inconsistency analysis (right half) is fully automated while ontology extraction (left half) is manual and performed only once. Inspired by the *soundness* (i.e., *no-false-positive*) in software testing with dynamic analysis [30, 31, 66], PurPliance is designed to maximize the precision of detection (i.e., a reported inconsistency should always be true positive), as opposed to maximizing the recall rate.

PurPliance addresses the following three technical challenges. **TC1 (Purpose clause extraction)**: Purpose clauses are written in lengthy and complex phrases, and hence it is difficult to determine their start and end in a sentence. PurPliance leverages neural *Semantic Role Labeling* (SRL) models [35, 62] that are capable of analyzing many more grammatical variations than prior work [7], to extract privacy statement parameters from the semantic arguments of data-practice predicates. Finally, PurPliance extracts uncompounded purposes from complex purpose clauses by analyzing their semantic/syntactic structures and decomposing the clauses into simpler predicate-object pairs and noun phrases. We organize the common purpose clauses extracted from a large collection of privacy policies into a hierarchical taxonomy that defines the relationships among different usage purposes.

TC2 (Data flow extraction): Extracting the purpose of data flows to/from each app is very challenging because the flows take place at a low data level and lack high-level semantics. PurPliance leverages recently-developed datasets and dynamic analysis techniques [33] to infer the purposes and the purpose-served entities of network data traffic from the transferred data and its context. The low-level purposes of data traffic are then mapped to higher-level data-usage purposes in our taxonomy of data purposes.

TC3 (Automated consistency analysis): Automatic detection of contradictory privacy policy statements and inconsistent network data flows requires automated reasoning of these concepts. We introduce the notion of *data-usage purpose* which comprises a purpose-served entity and a usage purpose, and is separated from data collection and sharing. We formalize privacy statements and data flows, and formulate a consistency model to analyze and detect policy contradictions and flow-to-policy inconsistencies.

The evaluation of our end-to-end contradiction detection demonstrates that PurPliance is able to detect contradictory sentence pairs in privacy policies with significantly higher precision and recall than PolicyLint [7], a state-of-the-art policy analysis technique. An in-depth analysis shows two main sources of these improvements: 1) semantic-argument analysis improves the extraction of privacy statement tuples and 2) data-usage purpose analysis enhances the expressiveness of the privacy statement tuples to reflect the policy sentences' semantics more accurately. This paper makes the following main contributions:

- *Automatic extraction and classification of data usage purposes in privacy policies.* We developed automatic extraction of purpose clauses based on semantic arguments of the data practice predicates (Sections 3.1). We introduced predicate-object pairs to extract simple purposes from a complex clause (Section 3.2). We studied data usage purposes in a large privacy policy corpus to construct a purpose taxonomy and develop automatic classifiers. To the best of our knowledge, this is the first large-scale study and classification of data usage purposes in privacy policies.
- *Formalization and automatic extraction of privacy statements and data flows with support for data-usage purposes.* We developed NLP-based automatic methods to extract privacy statements with data-usage purposes from policy sentences (Section 4). We adapted existing methods to extract data flows with data purposes from network data traffic (Section 5).
- *A formal consistency model with support for data-usage purposes.* We propose a formal model to detect contradictions in privacy policies and flow-policy inconsistencies between privacy policies and mobile apps' data collection (Section 6).
- *An end-to-end system* (called PurPliance, open sourced at [18]) that detects inconsistencies between the privacy policy and actual data collection of an app. *A corpus of 108 privacy policies* (publicly available at [18]), containing 5.9k sentences and 189 contradictory sentence pairs, was constructed to evaluate the end-to-end contradiction detection. The results show that PurPliance improves the precision from 19% to 95% and the recall from 10% to 50% compared to PolicyLint. An in-depth analysis shows that PurPliance extracts 88% more privacy statements in 45% more sentences with 9% higher precision than PolicyLint.
- *A large-scale study of policy contradictions and flow–policy inconsistencies in 23.1k Android apps* (Section 8). PurPliance found 29,521 potential contradictions in 18.14% of the policies and 95,083 inconsistencies in 69.66% of the apps, indicating the prevalence of inconsistencies of data-usage purposes in mobile apps.

2 RELATED WORK

Purpose Analysis in App Behavior. There has been a rich body of work to extract semantics of app behavior to identify potential leakage of sensitive information. Whyper [53], AutoCog [56] and CHABADA [23] analyze and assess the risks of an app's behavior (e.g., permission and API usage) in comparison with the app's description. FlowCog [52] extracts semantics of data flows from an app's GUI to analyze information leaks. NoMoATS [63] inspects the URL and HTTP headers to detect mobile network requests engaged in advertising and tracking. MobiPurpose [33] extracts and infers personal data types and purposes of their data collection from network traffic of Android apps, but it does not check whether the data-collection purposes are legitimate or not.

Privacy Policy Analysis. NLP and ML have been widely used for analyzing natural-language privacy policies. Privee [74] and Polisis [27] analyze privacy policies at the document- and paragraph-level to answer users' questions. However, both are limited by their coarse-grained analyses while our sentence- and phrase-level analyses provide more detailed and comprehensive results. PolicyLint [7] uses dependency parsing to extract privacy statements from policy documents but does not analyze purposes of data collection.

Bhatia *et al.* [11] extract common patterns of purposive statements from privacy policies and use semantic frames to analyze the incompleteness of privacy goals, which include the purposes of data practices [12]. Shvartzshnaider *et al.* [64] analyze information

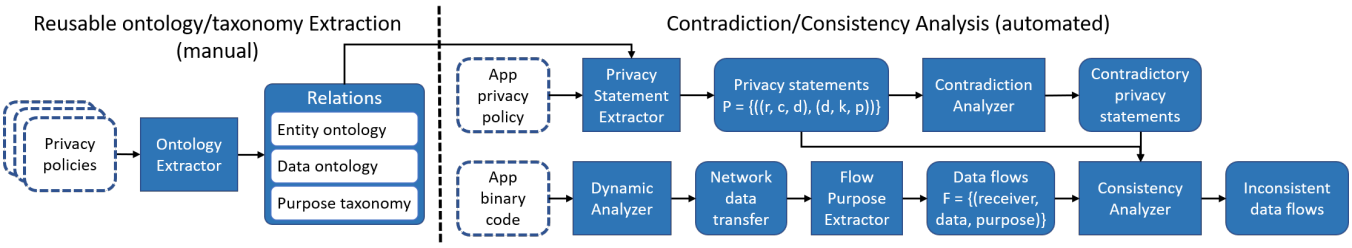

Figure 1: PurPliance system workflow. Dashed boxes indicate the system inputs.

Data Practice	Verbs
Sharing	disclose, distribute, exchange, give, lease, provide, rent, release, report, sell, send, share, trade, transfer, transmit
Collection	collect, gather, obtain, receive, record, store, solicit
Use	access, analyze, check, combine, connect, keep, know, process, save, use, utilize

Table 1: List of the SCoU verbs used by PurPliance.

flows in a limited set of privacy policies following the contextual integrity framework [48]. However, these semi-automated methods require the laborious manual efforts of experts or crowd workers.

Behavior-Policy Compliance Analysis of Mobile Apps. Analysis of the (in)consistencies between the actual behavior of mobile apps and their privacy policies has gained considerable interest in recent years. Prior work tracks the collection of users' personal data automatically via Android API calls [65, 73, 75], or via user inputs on app GUI [70]. PoliCheck [8] built upon PolicyLint [7] and the AppCensus dataset [9] improves the accuracy of detecting non-compliances in an app's data flows by taking into account the recipients of the personal data. However, PoliCheck does not consider the business purposes of the data flows. Several researchers focus on narrow app categories, such as paid apps [25] or apps targeting family users and children [50, 59, 60]. They are limited to specialized app categories while PurPliance is general and applicable to any type of apps.

Taxonomy of Privacy Purposes. OPP-115 dataset [71] includes 11 classes of data collection and sharing purposes that were manually created in a top-down fashion by law experts. In contrast, we created a hierarchical taxonomy of data-usage purposes by using neural text clustering with contextualized word embeddings to group similar purpose clauses in a large policy corpus. Despite a rich body of work on text clustering [3, 44], we are not aware of any other work that applies text clustering to the analysis of purposes in privacy policies.

3 EXTRACTION OF DATA USAGE PURPOSES

3.1 Extraction of Data Usage Purpose Clauses

3.1.1 Extraction of Data Practice Predicates and Semantic Arguments. PurPliance extracts the purposes of privacy practices by analyzing patterns of semantic arguments, syntactic structures (i.e., parts of speech and dependency trees) and a lexicon of data practices. It first finds data practice predicates (i.e., verbs) that express the action of a privacy practice event such as "collect" and "share". PurPliance iterates through the tokens of the sentence and extracts those words whose part-of-speech tags are a verb and whose lemmas are in a manually curated list of Sharing-Collection-or-Use (SCoU) verbs as given in Table 1.

We empirically identified common verbs in randomly selected privacy sentences to extend the SoC verbs in PolicyLint [7]. While PolicyLint only distinguishes between collection and sharing of data, we separate some *use* verbs. Although the *use* actions do not explicitly construct personal data flows, they still provide valuable information about data processing purposes. We added a verb to the SCoU list by surveying its usage in randomly selected sentences in our privacy policy corpus. Because every verb has multiple meanings, some of which are unrelated to data collection/sharing/use, there is a trade-off: naively adding verbs increases recall but reduces precision. Therefore, we select verbs that are frequently used to express data practices (i.e., in over 80% of 100 random sentences).

Given a data practice predicate, PurPliance analyzes its semantic arguments which are phrases that fill the meaning slots of the predicate and define its details. They answer questions such as "who?", "did what?", "to whom?", and "for which purpose?" of an event expressed by the predicate [35, 39]. Because arguments of the same event are consistent across varying syntactic forms, parameters of privacy statements (such as the receiver and data object) can be extracted accurately even though the same data practice event is expressed in multiple ways with varying grammars. An example of semantic arguments in varying expressions is given in Appendix A.

PurPliance uses Semantic Role Labeling (SRL), also called *shallow semantic parsing*, to recover the latent predicate-argument structures of sentences. SRL models are trained on corpora called *proposition banks* (PropBank) which contain labels of the semantic roles of sentences. In the corpora, such as OntoNotes 5.0 [57], a specific set of roles is specified for different senses of each verb. Some roles are numbered rather than named to make them more general (e.g., *Arg1* for *object* arguments) while many un-numbered modifier arguments represent the modification or adjunct meanings [14]. The definition of a role may vary with a verb's senses. For example, while *Arg2* typically denotes the instrument of a predicate, *Arg2* of certain data usage verbs like *use* and *store* indicates their purposes.

3.1.2 Extraction of Purpose Clauses. We identified semantic arguments that represent purposes based on their specifications in the CoNLL2012 corpus' verb sense frames [57]. The common arguments for purposes are *Argm-Prp* and *Argm-Pnc*, i.e., argument modifier purpose and purpose-not-cause, respectively. Table 2 presents some

Data Action	Sender	Receiver	Data	Purpose	Example
Sharing	Arg0	Arg2	Arg1	Argm-Prp or Argm-Pnc	[We]$_{\text{Arg0}}$ do not [share]$_{\text{V}}$ [your data]$_{\text{Arg1}}$ [with third parties]$_{\text{Arg2}}$ [for their purposes]$_{\text{Argm-Pnc}}$.
Collection	Arg2	Arg0			[We]$_{\text{Arg0}}$ [collect]$_{\text{V}}$ [passwords]$_{\text{Arg1}}$ [for authentication]$_{\text{Argm-Prp}}$.
Use	N/A	Arg0			[We]$_{\text{Arg0}}$ may [process]$_{\text{V}}$ [your contact information]$_{\text{Arg1}}$ [to send you promotions]$_{\text{Argm-Prp}}$.

Table 2: Mapping from semantic roles to privacy statement parameters. *V* denotes a predicate (i.e., verb).

examples. Besides the common purpose arguments, PurPliance analyzes additional arguments for certain predicates to identify their purposes, such as *Arg2* of *use* and *save*. The list of these predicate-specific purpose arguments is shown in Table 12 (Appendix A).

A verb may have multiple meanings, such as "save" which means either to save money or to collect (accrue) things. The latter meaning is more relevant in our context of data collection. We verified that arguments of different senses of the data practice verbs have the same meaning for the purpose of our privacy statement extraction, and hence we do not disambiguate the verb senses in this analysis.

We consider three forms of purpose clauses that either (1) start with "to" followed by a base-form verb, (2) start with "in order to" followed by a base-form verb, or (3) start with "for" followed by a gerund (a noun derived from a verb ending with *-ing*) or a noun. The first two forms are the standard identification of purpose clauses in English [34, 37, 46]. The third form is common in privacy policies, such as in "for providing services" or "for the purposes of ..."

3.2 Classification of Policy Purposes

3.2.1 Uncompounded Purpose Extraction. Since multiple simple purposes are commonly combined into complex purpose clauses, PurPliance decomposes them into simple single-purpose parts, called *uncompounded purposes*, similar to contextual sentence decomposition [10, 19] used to improve information extraction. Therefore, each complex purpose clause is simplified into a set of uncompounded purposes, each of which is represented by a predicate-object (PO) pair. A PO pair *(p,o)* consists of a predicate (verb) p that acts on an object o. For example, "to provide and improve our services" is decomposed into (*provide*, *our services*) and (*improve*, *our services*). Similarly, each noun phrase *np* can be converted to a PO pair with an empty predicate *('', np)*. For example, "for fraud prevention and service maintenance" produces "fraud prevention" and "service maintenance". Table 13 shows some PO-pair examples.

Each PO pair is extracted by first identifying the predicates and then their objects as its arguments. To extract a predicate, PurPliance finds words with a *verb* part of speech, excluding subsumptive relation verbs (e.g., *including* and *following*). Predicates also include past participles used as adjectives, such as "*personalized* content." The objects are then the noun phrases in each identified predicate's arguments. Similarly, PurPliance creates PO pairs whose predicates are empty and objects are the longest non-overlapping noun phrases extracted from the purpose clause by using a noun phrase extraction technique [26].

3.2.2 Purpose Taxonomy. We extracted uncompounded purpose clauses from a large collection of privacy policies and categorized them into semantically-similar groups to create a taxonomy of purposes. This process of creating a purpose taxonomy is different from data-object and entity ontologies [7] because privacy policies do not have subsumption expressions for purposes as commonly used for data types and entities, e.g., "personal information *includes* email address and name". First, from the privacy policy corpus, purpose clauses were extracted as described in Section 4. Purpose phrases with invalid prefixes (not beginning with "to", "for" or "in order to" + V) or empty PO pairs were filtered out. Uncompounded phrases were then created by concatenating the predicate and the object of each PO pair of the extracted purpose clauses. Finally, uncompounded purpose clauses with the number of occurrences greater than a threshold τ were selected to construct a taxonomy.

The uncompounded purpose clauses are grouped into semantically similar groups by using text clustering [44]. Each clause was converted into real-value vectors using *roberta-large-nli-stsb-mean-tokens*, a BERT-based sentence embedding model trained on semantic textual similarity datasets [58]. The vectors were grouped into γ clusters by K-means clustering [61]. The number of embedding groups was chosen heuristically by visualization using t-SNE [43] and by balancing the trade-off between granularity and complexity of the taxonomy.

We chose to use a small number of high-level groups to keep the taxonomy simple while still achieving the goal of detecting contradictions and inconsistencies. From 17k privacy policies, 392k uncompounded purpose clauses were extracted. 6,068 unique uncompounded purpose clauses were then selected using frequency threshold $\tau = 5$. This threshold was empirically chosen to remove noisy rare purpose clauses while shortening the t-SNE visualization time so that we can iteratively develop purpose-clusters without losing common purpose clauses. We conducted an iterative process of adjusting the number of classes and categorizing PO pairs to the selected classes until only a small number of PO pairs do not fit the taxonomy. $\gamma = 16$ was chosen for 15 clusters with a concrete purpose and 1 cluster with *Other* purpose. *Provide ad* and *Personalize ad* are separated for fine-grained classification. Providing ad indicates to only deliver, show, or provide advertising while personalizing ad indicates to customize, personalize, or tailor advertising. Since the purposes in the *Other* class are unrecognized purpose clauses, they do not have relationships (e.g., subsumption) with each other and are thus excluded from the consistency analysis.

Based on the economic activities of businesses [38], the γ low-level classes were further grouped into high-level categories: Production, Marketing, Legality, and Other categories. In the taxonomy, a low-level purpose is an instance of (i.e., has a subsumptive relationship with) the corresponding high-level purpose. For example, *Provide ad* is a *Marketing* purpose. In addition, we consider *Personalize ad* to be subsumed under *Provide ad* and *Personalize service*. If a service personalizes ads, then it provides ads, but not vice versa, because an ad can still be displayed without being personalized to the user's interest. The taxonomy is listed in Table 3's left half.

3.2.3 Data-Usage Purpose Classifier. PurPliance classifies purpose clauses by matching patterns of n-grams (e.g., words and

High-level	Low-level	Predicate Patterns	Object Patterns
Production	Provide service	provide, deliver	service, app, product
	Improve service	improve	
	Personalize service	personalize, customize	
		base	location service
	Develop service	track, detect	issue, bug
	Manage service	administer, manage	service, app, product
	Manage accounts	create, manage	account
	Process payments	process, complete	payment, transaction
	Security	detect, investigate, prevent	breach, fraud
		authenticate, verify	user, identity
Marketing	Customer communication	notify	user
		send	update
		resolve	inquiry
	Marketing analytics	analyze	usage, trend
	Promotion	send	promotion, reward
	Provide ad	provide, deliver	advertising, advertisement
	Personalize ad	personalize, target	
	General marketing		marketing
Legality	General legality	enforce	term, right
		comply	law
Other	Other purposes		

Table 3: Left half: high- and low-level purposes in the data usage purpose taxonomy; Right half: examples of patterns of the predicates and objects in purpose clauses.

bigrams) on lemmas of predicates and objects in the PO pairs of purpose clauses. We observe patterns that do not depend on the statement context, so matching such n-gram context-insensitive patterns provides precise classification. Moreover, PurPliance may classify one clause into multiple categories. For example, "provide personalized services" would be classified into *Provide service* and *Personalize service*.

To develop patterns and evaluate classification performance, we first extracted purpose clauses from all privacy policies and randomly divide them into training and test sets. 198,339 purpose clauses were extracted from our privacy policy corpus of 16.8k unique privacy policies. The training and test sets have 158,671 (80%) and 39,668 (20%) purpose clauses, respectively.

Patterns were developed on the training set which is disjoint from the test set. We randomly selected 1000 sentences in the training set and classified them until reaching a desirable coverage. The patterns covered 46% of the training set and 44% of the test set. The right half of Table 3 lists some example patterns on PO pairs.

To evaluate the classifier's precision, we randomly selected purpose clauses from the test set and classified them until each purpose class in the taxonomy contains at least 30 samples. The extracted purposes were then independently verified with the purpose taxonomy (Table 3) by two co-authors. Their disagreements were resolved via follow-up discussions. Of 510 randomly-selected samples in the test set, PurPliance achieved 97.8% precision on average. This high precision of the classifiers is due partly to the use of strict rule-based matching. The precision score of each purpose class is shown in Table 14 (Appendix C).

4 PRIVACY STATEMENT EXTRACTION

4.1 Definition of Privacy Statement

Each sentence in a privacy policy is formalized as a privacy statement which has two components: *Data Collection* which is the transfer of data to a receiver and *Data Usage* which represents the usage of the data and its purpose.

Definition 4.1 (Privacy Statement)**.** *A privacy statement is a pair* (dc, du) *where dc (du) represents data collection (usage).* $dc = (r, c, d)$ *denotes whether or not a receiver* r *collects (*$c \in \{collect, not\_collect\}$*) a data object* d*.* $du = (d, k, p)$ *represents whether or not data* d *is used for (*$k \in \{for, not\_for\}$*) an entity-sensitive data usage purpose* p*.*

The data usage can be a special *None* value when the statement does not specify any purpose for the data collection or PurPliance cannot extract the purpose from a sentence. While a privacy statement can be represented as a flat 5-tuple (r, c, d, k, p), we explicitly separate data collection *dc* from data usage *du* to distinguish the source of a contradiction which is either *dc* or *du*. Furthermore, our contradiction analysis can use hierarchical checking that has a smaller number of rules than that for the high-dimensional flat representation. Moreover, the 5-tuple representation also suffers from a large number of relationships between two tuples which increase exponentially with the number of tuple dimensions. Separating the data usage from data collection creates a constraint that if c = not_collect, then *du* should be *None* because the data object d cannot be used without collecting it first.

Entity-Sensitive Data Usage Purposes. We define entity-sensitive data usage purposes as follows to capture the meaning of statements that mention whether the data is used for the purposes of the app itself or a third party. For example, "for third parties' own marketing purposes" is represented as a pair *(third party, marketing)*.

Definition 4.2 (Entity-Sensitive Data Usage Purpose)**.** *An entity-sensitive purpose of data usage is a pair* (e, q)*, where* e *is the entity whose purpose is served, called purpose-served entity, and* q *is a data usage purpose.*

As an example, "third parties do not collect device identifiers for their advertising purposes" will be translated into a statement *(dc=(third party, collect, device ID), du=(device ID, not_for, (third party, advertising))*. We assume third parties still collect device IDs but the data is not used for third parties' advertising purposes. Because of "their" word, we also assume the data serves third parties' purposes.

Compared to PolicyLint, PurPliance adds a new data usage representation *du*, uses a representation of data usage purpose and has a more complete interpretation of privacy sentences. While the *dc* component contains the same parameters as in PolicyLint, PurPliance uses a different interpretation of data collection in privacy policy sentences. Given the above sentence, PolicyLint creates *(third party, not_collect, device ID)* but it implies absolutely no collection of device IDs and would flag any other statements about the collection of a related data type.

4.2 Extraction of Statement Parameters

PurPliance extracts phrases that correspond to the parameters of privacy statements from a sentence in 3 steps: (1) identify data practice predicates (verbs), (2) extract the semantic arguments of each predicate and (3) map these arguments to the parameters.

Receiver Extraction. The receiver and sender of a data practice are determined by either *Arg0* or *Arg2*, depending on the action type (i.e., *collection*, *use*, or *sharing*). Since *Arg0* and *Arg2* are typically

Rule	Extracted span*	Created privacy statements
T_1	*(r, not_collect, d, None)*	*((r, not_collect, d), None)*
T_2	*(r, not_collect, d, p)*	*((r, collect, d), (d, not_for, p))*
T_3	*(s, share, r, d, None)*	*((s, collect, d), None)* and *((r, collect, d), None)*
T_4	*(s, not_share, r, d, None)*	*((s, collect, d), None)* and *((r, not_collect, d), None)*
T_5	*(s, share, r, d, p)*	*((s, collect, d), None)* and *((r, collect, d), (d, for, p))*
T_6	*(s, not_share, r, d, p)*	*((s, collect, d), None)* and *((r, collect, d), (d, not_for, p))*

Table 4: Privacy statements created from extracted text spans. * *text span = (sender, action, receiver, data, purpose).*

the actor and the beneficiary of an action, if the action is collection, *Arg0* is the receiver and *Arg2* is the sender of the data object. Similarly, these roles are swapped if the action is sharing. In the case of data-using actions, there is no sender and *Arg0* represents the entity that uses the data. The mapping from the arguments to the sender/receiver is shown in Table 2. The first or third party can also be mentioned implicitly in a sentence depending on the type of the data practice. For example, in "we will not share your sensitive data," the missing receiver is inferred as an *implicit third party* since the type of data action is *sharing*. When a verb is a clausal complement (i.e., its dependency is *xcomp*) and the receiver is an object pronoun, it would be converted to a subject pronoun. For example, in "you authorize us to collect your personal data to provide the services," Arg0 of *collect* is *us* which is then converted to *we*.

Conversion from Extracted Spans to Privacy Statements. Privacy statements are generated from extracted spans by using transformation rules as listed in Table 4. Rules T_1 and T_2 convert spans with a collection verb while rules T_3–T_6 convert spans with a sharing verb. A rationale behind rules T_3–T_6 is that the data collection and sharing are observed only at the client side (i.e., the app), and hence the data collection or sharing on the server side is unknown. In particular, rule T_3 assumes the sender may have the data before sharing it. Similarly, rule T_4 means while the sender does not share data, it may still collect the data. Rules T_5 and T_6 mean the receiver may still collect data, but the data should not be used for the purpose *p*. For example, given "we do not provide your personal data to third parties for their own marketing purposes," we interpret this statement as personal data can be transferred to third-party service providers, but does not serve the third parties' purposes.

Action Sentiment Extraction. The sentiment of a data practice can be either positive or negated and indicates whether the data action is performed or not, respectively. The sentiment is determined by checking the presence of the negation argument *Argm-Neg*. If the predicate has no *Argm-Neg*, `PurPliance` analyzes its dependency tree to determine its negation using the method in PolicyLint [7]. For example, in "we never sell your data," *sell* has a negated sentiment because it has a negation argument *never*. However, the negation of *use*-verbs does not generate *not_collect* because "app does not use data A" does not mean the app does not collect data A.

Data Object Extraction. `PurPliance` extracts the text spans of the objects of the privacy practice actions using SRL and extracts data object noun phrases using named entity recognition (NER) [35]. First, argument *Arg1* is mapped to the Data component since it is the object of a verb across data practice action types. Second, the verb argument is then further refined by using NER which is a common technique used to extract data objects [7]. For example, given "we may use your name and street address for delivery", NER extracts "your name" and "street address" from the corresponding argument identified by SRL.

Purpose-Served Entity Extraction. Although it is more accurate to determine the purpose-served entities by performing co-reference resolution [35], `PurPliance` uses keyword matching to extract purpose-served entities. `PurPliance` leverages an observation that "their" commonly means third parties because data-practice statements in privacy policies are frequently between first-party/users and third parties, such as in the sentence "third parties may not use personal data for their own marketing purposes." Similarly, "our" in purpose clauses commonly refers to first parties. If no such entities were found, the purposed-served entity is set to "any party".

Purpose analysis allows more accurate interpretations of certain sharing statements. In particular, when user data is used for "monetary" or "profitable" purposes or when the data practice is to "lease", "rent", "sell" and "trade" the user's data, we interpret that the data is shared for a third party's purposes. To reduce false positives, we only include the *Marketing – Provide ad* purpose which is the most common. When an advertiser collects a data object for advertising purposes, the purpose-served entity is also set to the advertiser.

Exception Clauses. Given a sentence that includes an exception clause which does not contain data objects or entities, if the privacy statement extracted from the sentence has a negated sentiment, `PurPliance` changes the sentiment of the privacy statement to be positive. For example, "we do not share your personal data with third parties for their marketing purposes without your consent" produces text spans *(we, share, third party, your personal data, (their, marketing))*. This exception clause handling is similar to PolicyLint.

`PurPliance` generates additional privacy statements in certain cases when a sentence contains exceptions about purposes. If the sentence is negated and contains "other than [purpose clause]," the data will not be used for other high-level purposes. Excluding other high-level purposes which are semantically non-overlapping produces fewer false positives than excluding other low-level purposes.

Similarly, `PurPliance` creates opposite-sentiment privacy statements for other purposes if the sentence contains "for [purpose clause] only," or "only for [purpose clause]." `PurPliance` also excludes the data usage for the purposes of third parties given purpose-restrictive phrases such as "only for internal purposes." Although many third parties' purposes can be considered to be outside of "internal purposes", we exclude only *Marketing – Provide ad*, which is the most common, to reduce false positives.

5 DATA FLOW EXTRACTION

5.1 Data Purpose Analysis

PurPliance re-implements MobiPurpose approaches [33] to infer the data types and usage purposes from mobile apps' network traffic (Sections 5.3 and 5.4). The data types and purposes of app–server communication are inferred from the content of the data sent to the server, the destination and the app description. While the semantics of data is vague, resource names (e.g., variables or server names) are assumed to clearly reflect their intentions [33, 63] as it is necessary for effective software engineering [47] and especially important for server names shared by multiple parties. The rationale of feature selection and system design is discussed at length in [33].

Since only the dataset of MobiPurpose [33] is available, we reproduce its inference and adapt the labels to the purposes in our purpose taxonomy. We assume the human annotators of MobiPurpose dataset correctly labeled the purposes of data, so the high agreement of ML with human annotators means that ML predicts the purposes of data with a high probability.

5.2 Data Flow Definition and Extraction

Definition 5.1 (Data Flow)**.** *A data flow is a 3-tuple* (r, d, p) *where a recipient* r *collects a data object* d *for an entity-sensitive purpose* p *and* $p = (e, q)$ *where* e *is the purpose-served entity and* q *is a data-usage purpose.*

App requests are commonly structured in key–value pairs [67], so each structured data sent to the server is decomposed into multiple key–value pairs $\{kv_l\}$. Therefore, each request or response between app app_i and end-point url_j corresponds to a set of low-level flows $F = \{f_k | f_k = (app_i, url_j, kv_l)\}$. The data type d, purpose-served entity e and usage purpose q are then inferred from F.

PurPliance distinguishes first and third parties to determine the purpose-served entity e by analyzing the receiver r and the inferred data-usage purposes q. For example, an app's "supporting" services like content delivery networks (CDNs) use data for the app's *first-party* purposes rather than for another party's. While there are many combinations of r (e.g., *First-party* or *Third-party*) and q (one of 5 purposes, Table 6), to avoid false positives, we conservatively set *e=Advertiser* only when *r=Advertiser* and *q=Provide ad* or *Personalize ad* (i.e., an advertiser uses collected data for its advertising purposes). For other cases, such as when an app uses "supporting" third-party services (i.e., *r=Third-party* and *q=Provide service*), data usage still serves the first-party's purposes, and hence we set *e=First-party*. The receivers of data flows are resolved by checking the data's destination URL with the package name, the privacy policy URL and well-known analytics/advertisement lists [2]. Note that purpose-served entity e is not supported by MobiPurpose.

PurPliance uses dynamic analysis to exercise the apps and capture their network data traffic. It has 2 advantages over static analysis: (1) real (not just potential) execution, hence reducing false positives, and (2) destination of data, which can be determined dynamically on the server side.

By analyzing purposes of data flows, PurPliance can distinguish more fine-grained intentions of data usage than entity-only approaches like PoliCheck [8]. In particular, the 1st party can collect data for its own marketing purposes. For example, *Wego Flights* app sends a client ID to its own server at *srv.wego.com* with the request path */analytics/visits*, so the collection of user ID can be inferred to be for the app's purpose of Marketing Analytics. The sent data's semantics is especially useful to distinguish data transfer to a business partner of the app which is not a popular advertisement network or analytic service provider.

Data Type	Precision	Recall	F1	Support
Identifiers	0.98	0.92	0.95	141
Geographical location	0.98	0.94	0.96	67
Device information	0.98	0.89	0.93	45
Network information	1.00	0.92	0.96	26
User profile	0.89	1.00	0.94	16
Average	0.97	0.93	0.95	59

Table 5: Data type extraction performance.

5.3 Data Type Extraction

Using the corpus from MobiPurpose [33] which contains manually-annotated data types for key–value pairs of apps' network traffic, we identified patterns of the key-values for each data type. The corpus has 5 high-level data types (listed in Table 5) that are common in app data communication: identifiers (i.e., hard/software instance and advertising IDs), network information (e.g., types of network), device information (e.g., device types and configurations), location (e.g., GPS coordinates) and user account information (e.g., user name, password and demographics). MobiPurpose dataset does not distinguish types of ID (i.e., advertising, hardware and instance ID) which are frequently used by developers for overlapping purposes. The distinction can be achieved by finer-grained data type labels. However, developing such a dataset is beyond this paper's scope.

Data Type Features. There are 2 types of patterns: special strings and bag-of-words. The key–value strings are first matched by special-string patterns which comprise unigrams, bigrams, regular expressions and bags of words. If no match is found, an English Word Segmentation model [32] was used to segment the key–value pairs into separate words and construct a bag of words. For example, "sessionid" is separated into *session* and *id*. The occurrence of the word *id* indicates this is an identifier. These patterns become 6-component feature vectors where each component is whether there is any matched pattern or not. The last component is set to 1 if there is no matched pattern for the 5 data types. We tried 4 types of classifiers (Logistic Regression (LR), Multi-Layer Perceptron (MLP), Random Forest (RF) and Support Vector Machine (SVM)) to classify these features. The best-performing classifier is found to be Random Forest with 200 estimators.

Performance Evaluation. The corpus is randomly divided into a development set (80%) for developing string patterns and a test set (20%) for evaluating 5 data-type classifiers. We remove types with too few (i.e., less than 20) samples. The classifiers achieve 95% F1 score with 97% precision and 93% recall rates on average. The precision is more than 89% on all data types. The high accuracy indicates the regularity in key and values which were programmatically produced by the apps. The lowest recall is of the *device information* data type because the classifier misclassifies some samples which

Purpose Class	Prec.	Rec.	F1	Sup.
Production - Provide service	0.76	0.81	0.78	15
Production - Personalize service	0.85	0.66	0.74	18
Production - Security	0.81	0.73	0.77	16
Marketing - Provide ad	0.86	0.86	0.86	76
Marketing - Marketing analytics	0.77	0.85	0.81	72
Average	0.81	0.78	0.79	39

Table 6: Purpose prediction performance on the data flows in the test set. The total number of samples is 1413. The classifiers are tuned for the extraction precision. The metric columns are Precision/Recall/F1/Support in this order.

look like device IDs, such as *clientId: Huawei+Nexus+6P*, but are actually a device model. The detailed performance results of the classifiers are provided in Table 5.

5.4 Data Traffic Purpose Inference

Data Usage Purpose Features. `PurPliance` uses the same features as those in MobiPurpose to predict the purposes of each key–value pair in the transferred data. There are 6 features in 3 groups based on the destination URL, sent data and app package name. The first group of features are based on the usage intention embedded in the semantics of the destination URL and sent data which have a form of *scheme* : *//host/path*. The second feature group encodes the characteristics of the data types in the sent data such as the number of key-value pairs. The third feature group shows the relation between the app and the server. For example, the data sent to *cbc*2015.*prod*1.*sherpaserv.com/services* by app *com.sherpa.cbc*2015 is likely to the app's server. They are encoded in 291-dimensional vectors (Table 15 in Appendix E).

Purpose-Classification Dataset. The purpose classification models were trained on MobiPurpose corpus [33] which contains Android apps' network data traffic. We obtained a total of 1413 samples. The data types and purposes of each key–value pairs contain labels created by 3 experts. We aggregated purpose labels into a single purpose label by using majority votes, following the method in the original paper [33]. Specifically, a sample is classified as a purpose p as the most common label from the annotators. The dataset has 24 categories in MobiPurpose taxonomy. We manually mapped them to 7 classes in `PurPliance` (as shown in Table 17, Appendix G). The final 5 purpose classes are listed in Table 6.

Performance Evaluation. Similar to data type classification, we experimented 4 types of machine learning models: LR, MLP, RF and MLP. The MLP uses ReLU activation and Adam optimizer with a fixed learning rate of 10^{-5}. The Random Forest has 200 estimators. We used random search for the hidden layers of 2-layer MLP, regularization strength (C) of Linear Regression models with range 0.1–10 estimators (range 100–200) for Random Forest. The evaluation was done on the dataset using 10-fold cross validation. Similar to [33], we removed purpose classes that have too few (i.e., less than 20) samples, such as the Other purpose class.

The average F1 score is 79% (81% precision and 78% recall). *Provide ad* and *Marketing analytics* have the highest F1 scores of 86% and 81%, respectively. The lowest F1 score is of the *Personalize service* class since it is challenging to distinguish this class from other classes such as *Provide service*. The results are given in Table 6.

We perform an ablation study to evaluate the effectiveness of the features used to predict the purposes. The results (listed in Table 16 in Appendix E) show that the type of the transferred data is the most effective feature that improves the F1 score by 4%. The number of key–value pairs also improves F1 by 2% since there is a correlation between this feature and the data purposes (e.g., analytic services often collect more key–value pairs (10+) than the rest [33]).

6 CONSISTENCY ANALYSIS

This section formalizes the detection of purpose inconsistencies within privacy policies (called *policy contradictions*) as well as those between the policies and the actual data collection and sharing behavior of the corresponding apps (called *flow-policy inconsistencies*).

6.1 Semantic Relationships

Each parameter in a privacy statement is mapped to an ontology (e.g., data object ontology and purpose taxonomy) which defines relationships among the terms used. We extend the semantic equivalence, subsumptive relationship and semantic approximation of PoliCheck [8] to data-usage purposes as listed in Table 7. R_1 is defined in Definition 6.4, $R_2 - R_4$ are defined in Definition 6.5, and $R_5 - R_9$ are defined in Theorem 6.6 (proved in Appendix D).

Definition 6.1 (Semantic Equivalence)**.** *$x \equiv_o y$ means that x and y are synonyms, defined under an ontology o.*

Definition 6.2 (Subsumptive Relationship)**.** *Given an ontology o represented as a directed graph in which each node is a term and each edge points from a general term y to a specific term x included in y (i.e., x "is a" instance of y), $x \sqsubset_o y$ means there is a path from y to x and $x \not\equiv_o y$. Similarly, $x \sqsubseteq_o y \Leftrightarrow x \sqsubset_o y \vee x \equiv_o y$ and $x \sqsupset_o y \Leftrightarrow y \sqsubset_o x$.*

Definition 6.3 (Semantic Approximation)**.** *The semantic approximation relationship between two terms x and y, denoted as $x \approx_o y$, is true if and only if $\exists z$ such as $z \sqsubset_o x \wedge z \sqsubset_o y \wedge x \not\sqsubseteq_o y \wedge y \not\sqsubseteq_o x$.*

Definition 6.4 (Purpose Equivalence)**.** *Two data-usage purposes are semantically equivalent $(e_i, q_i) \equiv_\pi (e_j, q_j)$ if and only if there exist ontologies ϵ and κ such that $e_i \equiv_\epsilon e_j \wedge q_i \equiv_\kappa q_j$.*

Definition 6.5 (Purpose Subsumption)**.** *$(e_i, q_i) \sqsubset_\pi (e_j, q_j)$ if and only if there exist ontologies ϵ and κ such that $e_i \sqsubset_\epsilon e_j \wedge q_i \equiv_\kappa q_j$ or $e_i \equiv_\epsilon e_j \wedge q_i \sqsubset_\kappa q_j$ or $e_i \sqsubset_\epsilon e_j \wedge q_i \sqsubset_\kappa q_j$.*

Theorem 6.6 (Purpose Semantic Approximation)**.** *Given two data-usage purposes $p_i = (e_i, q_i)$ and $p_j = (e_j, q_j)$, there exist ontologies ϵ, κ, and π such that*

(1) $e_i \equiv_\epsilon e_j \wedge q_i \approx_\kappa q_j \Rightarrow p_i \approx_\pi p_j$,
(2) $e_i \sqsubset_\epsilon e_j \wedge q_i \approx_\kappa q_j \Rightarrow p_i \approx_\pi p_j$,
(3) $e_i \approx_\epsilon e_j \wedge q_i \equiv_\kappa q_j \Rightarrow p_i \approx_\pi p_j$,
(4) $e_i \approx_\epsilon e_j \wedge q_i \sqsubset_\kappa q_j \Rightarrow p_i \approx_\pi p_j$, and
(5) $e_i \approx_\epsilon e_j \wedge q_i \approx_\kappa q_j \Rightarrow p_i \approx_\pi p_j$

Relation	$e_i \cdot e_j$	$q_i \cdot q_j$	$p_i \cdot p_j$
R_1	$e_i \equiv_\epsilon e_j$	$q_i \equiv_\kappa q_j$	$p_i \equiv_\pi p_j$
R_2	$e_i \equiv_\epsilon e_j$	$q_i \sqsubset_\kappa q_j$	$p_i \sqsubset_\pi p_j$
R_3	$e_i \sqsubset_\epsilon e_j$	$q_i \equiv_\kappa q_j$	$p_i \sqsubset_\pi p_j$
R_4	$e_i \sqsubset_\epsilon e_j$	$q_i \sqsubset_\kappa q_j$	$p_i \sqsubset_\pi p_j$
R_5	$e_i \equiv_\epsilon e_j$	$q_i \approx_\kappa q_j$	$p_i \approx_\pi p_j$
R_6	$e_i \sqsubset_\epsilon e_j$	$q_i \approx_\kappa q_j$	$p_i \approx_\pi p_j$
R_7	$e_i \approx_\epsilon e_j$	$q_i \equiv_\kappa q_j$	$p_i \approx_\pi p_j$
R_8	$e_i \approx_\epsilon e_j$	$q_i \sqsubset_\kappa q_j$	$p_i \approx_\pi p_j$
R_9	$e_i \approx_\epsilon e_j$	$q_i \approx_\kappa q_j$	$p_i \approx_\pi p_j$

Table 7: Data-usage purpose relationships. $p_i = (e_i, q_i)$ and $p_j = (e_j, q_j)$. $\cdot$ denotes a relationship placeholder. $R_1 - R_4$ are definitions, $R_5 - R_9$ are theorems.

Rule	Logic	Example
C_1	$d_k \equiv_\delta d_l \wedge p_m \equiv_\pi p_n$	(Device ID, k, Advertising) (Device ID, ¬k, Advertising)
C_2	$d_k \equiv_\delta d_l \wedge p_m \sqsubset_\pi p_n$	(Device ID, k, Advertising) (Device ID, ¬k, Marketing)
C_3	$d_k \sqsubset_\delta d_l \wedge p_m \equiv_\pi p_n$	(Device ID, k, Advertising) (Device info, ¬k, Advertising)
C_4	$d_k \sqsubset_\delta d_l \wedge p_m \sqsubset_\pi p_n$	(Device ID, k, Advertising) (Device info, ¬k, Marketing)
C_5	$d_k \sqsupset_\delta d_l \wedge p_m \sqsubset_\pi p_n$	(Device info, k, Advertising) (Device ID, ¬k, Marketing)
C_6	$d_k \equiv_\delta d_l \wedge p_m \approx_\pi p_n$	(Device ID, k, Advertising) (Device ID, ¬k, Personalization)
C_7	$d_k \sqsubset_\delta d_l \wedge p_m \approx_\pi p_n$	(Device ID, k, Advertising) (Device info, ¬k, Personalization)
C_8	$d_k \sqsupset_\delta d_l \wedge p_m \approx_\pi p_n$	(Device info, k, Advertising) (Device ID, ¬k, Personalization)
C_9	$d_k \approx_\delta d_l \wedge p_m \equiv_\pi p_n$	(Device ID, k, Advertising) (Tracking ID, ¬k, Advertising)
C_{10}	$d_k \approx_\delta d_l \wedge p_m \sqsubset_\pi p_n$	(Device ID, k, Advertising) (Tracking ID, ¬k, Marketing)
C_{11}	$d_k \approx_\delta d_l \wedge p_m \sqsupset_\pi p_n$	(Device ID, k, Marketing) (Tracking ID, ¬k, Advertising)
C_{12}	$d_k \approx_\delta d_l \wedge p_m \approx_\pi p_n$	(Device ID, k, Advertising) (Tracking ID, ¬k, Personalization)
N_1	$d_k \equiv_\delta d_l \wedge p_m \sqsupset_\pi p_n$	(Device ID, k, Marketing) (Device ID, ¬k, Advertising)
N_2	$d_k \sqsubset_\delta d_l \wedge p_m \sqsupset_\pi p_n$	(Device ID, k, Marketing) (Device info, ¬k, Advertising)
N_3	$d_k \sqsupset_\delta d_l \wedge p_m \equiv_\pi p_n$	(Device info, k, Advertising) (Device ID, ¬k, Advertising)
N_4	$d_k \sqsupset_\delta d_l \wedge p_m \sqsupset_\pi p_n$	(Device info, k, Marketing) (Device ID, ¬k, Advertising)

Table 8: Logical forms of logical contradictions (C) and narrowing definitions (N). k and $\neg k$ abbreviate *for* and *not_for*, respectively. The data flow has data type f_d = *IMEI* and purpose f_q = *Personalize ad*.

6.2 Policy Contradictions

Definition 6.7 (Privacy Statement Contradiction)**.** *Two privacy statements $t_k = (dc_k, du_k)$ and $t_l = (dc_l, du_l)$ are said to contradict each other iff either dc_k contradicts dc_l or du_k contradicts du_l.*

PurPliance's consistency analysis comprises two steps. Using the Definition 6.7 of contradiction between two privacy statements, it checks the consistency of *dc* and *du* tuples in this order. The consistency of $dc_k = (r_k, c_k, d_k)$ and $dc_l = (r_l, c_l, d_l)$ is analyzed by a Data Collection consistency model. PurPliance leverages the PoliCheck consistency model in this analysis. However, the PoliCheck consistency model cannot check the two policy statements if both have a positive sentiment (i.e., $c_k = c_l$ = collect) or

	X does not collect Y	X collects Y
X does not collect Y for Z	Consistent	Consistent
X collects Y for Z	Contradictory	Consistent

Table 9: Privacy-statement comparison when one of the statement has no data usage purpose specified ($du = None$).

the two receivers do not have either a subsumptive or semantic approximation relationship. In such cases, since no contradiction was detected, PurPliance checks the consistency of data usage statements du_k and du_l using a Data Usage consistency model. We extend the PoliCheck model [8] for data usage purposes as follows.

The contradiction conditions and types of two data usage tuples $du_k = (d_k, for, p_m)$ and $du_l = (d_l, not\_for, p_n)$ are listed in Table 8. There are 16 cases and 2 types of contradictions: *logical contradictions* (C_1–C_{12}) and *narrowing definitions* (N_1–N_4). Logical contradictions occur when du_l states the exclusion of a broader purpose from data usage while du_k states the usage for a purpose type in a narrower scope. On the other hand, narrowing definitions have the not-for-purpose statement (where $k = not\_for$) in a narrower scope than their counterparts. Narrowing definitions may confuse readers and automatic analysis when interpreting the privacy statements, especially when the two statements are far apart in a document.

When two privacy statements are compared, if one of them has no data-usage purpose specified (i.e., $du = None$), PurPliance flags a contradiction only if they have forms $((r_k, not\_collect, d_k), None)$ and $((r_l, collect, d_l), (d_l, for, p_l))$, i.e., the positive-sentiment statement has $k_l = for$. Following this rule, "X does not collect Y" does not contradict "X does not collect Y for Z" as they are translated to *((X, not_collect, Y), None)* and *((X, collect, Y), (Y, not_for, Z))*, respectively. Table 9 lists the cases of this rule.

Example 2. Given two statements: "we use your personal data only for providing the App" and "advertisers may use your device ID to serve you with advertisements," a contradiction is detected as follows. Due to the keyword *only for*, PurPliance excludes third parties' *Marketing* purposes that are not for *providing* the app and translates the first sentence to 1 positive and 1 negated statement: s_1^1 = *(we, collect, personal data), (personal data, for, (anyone, Provide service))*, s_1^2 = *(third party, collect, personal data), (personal data, not_for, (third party, Marketing))*. The second sentence is translated to s_2 = *(advertiser, collect, device ID), (device ID, for, (advertiser, Provide ad))*. Since *device ID* $\sqsubset$ *personal data*, *advertiser* $\sqsubset$ *third party* and *Provide ad* $\sqsubset$ *Marketing*, the first sentence's negated statement s_1^2 contradicts s_2 of the second sentence under rule C_4. PolicyLint will not flag these sentences because it considers only the collection tuples which are all positive sentiments in these sentences.

6.3 Flow Consistency Analysis

Definition 6.8 (Flow-relevant Privacy Statements)**.** *A privacy statement $t_f = ((r_t, c_t, d_t), (d_t, k_t, (e_t, q_t)))$ is relevant to a flow $f = (r, d, (e, q))$ (denoted as $t_f \simeq f$) if and only if $r \sqsubseteq_\rho r_t \wedge d \sqsubseteq_\delta d_t \wedge e \sqsubseteq_\epsilon e_t \wedge q \sqsubseteq_\kappa q_t$. Let T_f be the set of flow-f-relevant privacy statements in the set of privacy statements T of a privacy policy, then $T_f = \{t \mid t \in T \wedge t \simeq f\}$.*

Definition 6.9 (Flow-to-Policy Consistency). *A flow f is said to be consistent with a privacy policy T iff $\exists t \in T_f$ such that $c_t = collect \wedge k_t = for$ and $\nexists t \in T_f$ such that $c_t = not\_collect \vee k_t = not\_for$.*

A data flow is inconsistent with a privacy policy if the Flow-to-Policy Consistency condition is not met. For each flow extracted from app behavior, PurPliance first finds the flow-relevant privacy statements T_f and classifies the flow as consistent or inconsistent using the above definitions. Although finer-grained consistency types can be used, such as Clear and Ambiguous disclosures as in PoliCheck, we leave it as future work. For brevity, the definitions only include cases where data-usage purposes are specified. The conditions on purposes are not checked if the data purpose is unspecified (i.e., *du=None*).

Example 1 creates a privacy statement *((third party, collect, personal_data), (personal_data, not_for, (third party, Marketing)))*. Transferring the user device IMEI number to an advertiser's server creates a data flow *f=(advertiser, IMEI, (advertiser, Provide ad))*. Because *IMEI* $\sqsubset$ *personal_data* (via *device_identifier*), *advertiser* $\sqsubset$ *third party*, and *Provide ad* $\sqsubset$ *Marketing* (relationship in the purpose taxonomy), the data flow is inconsistent with the privacy statement.

7 SYSTEM IMPLEMENTATION

Semantic and Syntactic Analysis. PurPliance uses a neural SRL model [4, 62] trained on OntoNotes 5.0 [55, 54, 57], a large-scale corpus with 1.7M English words of news, conversations and weblogs and 300K proposition annotations. Each token is encoded into vectors depending on its context by using BERT-base-uncased contextualized word embeddings [17, 69]. Spacy with *en_core_web_lg* language model [20] was used for syntactic analysis and dependency parsing. Analyzing 16.8k privacy policies took 2 hours on 1 machine equipped with 2 Nvidia Titan Xp GPUs.

Data Object and Entity Ontologies. The consistency analysis logical rules require all entities and objects to be mapped into ontologies to check their subsumptive relationships. PurPliance extends the data object and entity ontologies based on PoliCheck to check their subsumptive relationship. Similar to the addition of SCoU verbs, we only add data objects and entities that are frequently used in data-practice statements to avoid noise from those used in unrelated sentences. PurPliance extracts data objects and entities by using a domain-adapted NER model trained on PolicyLint's dataset of 600 manually-annotated sentences (see Appendix H for details).

Policy Crawler and Preprocessor. We developed a crawler and preprocessor to collect the privacy policies of Android apps. Its implementation is described in Appendix F.

Network Data Traffic Collection. PurPliance used a tool based on the VPN server API on Android [24] to capture apps' HTTP(S) traffic which is the most common protocol in app–server communication [21]. A system certificate was installed on rooted phones for capturing encrypted traffic. Each app was exercised with human-like inputs generated by deep-learning-based Humanoid [41], built atop Droidbot automation tool [40]. For each app, the experiment ran for at most 5 min and stopped if there was no traffic generated for more than 2 min. These timeouts were empirically determined for a good trade-off between data coverage and the number of apps that we want to explore. We used 5 smartphones with Android 8.

8 EVALUATION

8.1 Data Collection

App Selection. We first selected the top 200 free apps for each of 35 categories on Google Play Store, excluding Android Wear and second-level Game categories [1]. This step resulted in 6,699 unique apps. Second, from a collection of 755,879 apps crawled from Google Play Store in May 2020, we randomly selected additional 28,301 apps that are different from the top apps in the first step and have been updated since 2015. To this end, 35k unique apps were selected. After removing apps with an invalid privacy policy, our final app corpus comprises 23,144 apps with a valid privacy policy.

Privacy Policy Corpus. We create a policy corpus as follows. We removed 6,182 duplicate policies from apps that share the same policy from the same developer. To reduce noise from titles (such as policy section titles), sentences with title-cased or all capitalized words or with less than 5 tokens are removed. Our final privacy policy corpus has 16,802 unique policies with 1.4M sentences. The categories with the most and least apps are Game (3,889 apps/2,797 policies) and Libraries & Demo (166 apps/121 policies), respectively. Fig. 5 (Appendix I) shows their distribution over app categories.

Capturing Network Traffic. We capture the traffic of only the apps which have a valid policy to analyze the app-flow consistency. We intercepted 3,652,998 network requests of 18,689 apps over 33 days. Among those, we discarded traffic with empty-body requests or not from apps with valid policies and apps which became unavailable from Play Store at the time of testing. The final dataset has 1,727,001 network requests from 17,144 unique apps. The number of apps that generated traffic is lower than the selected apps because they either work offline or our automated input generation did not generate any input which triggered any requests to the servers, or the apps require login preventing our tool from using the service. These apps contacted 19,282 unique domains (164,096 unique end-point URLs) and sent 24,918,567 key-value pair data to remote servers. The distributions of network data requests across domains and app categories are described in Appendix J.

8.2 Privacy Statement and Flow Distributions

PurPliance extracts 874,287 privacy statements from 142,231 sentences in 15,312 policies (93.6% of 16,362 apps with data flows extracted). Of these, 225,718 (25.8%) statements from 43,421 (30.5%) sentences contain extracted purpose clauses. PurPliance recognized 112,652 privacy statements with a non-Other purpose class. The most common purposes are *Provide Service* and *Improve Service* which appear on 72.6% and 59.6% of the apps' policies, respectively. Fig. 2 shows the distribution of privacy statements' purposes.

Using the models developed in Section 5, 701,427 unique data flows from 16,362 apps were extracted. Each data flow comprises a single key–value pair in the captured traffic of each app. 432,078 (61.2%) have a non-Other purpose and 282,984 (40.3%) have both non-Other purpose and data type. The Other class is for data types or purposes which our classifier was unable to infer such information as a key–value of encrypted data. The most common data types are Device Information and Identifiers which appear in 95.7% and 87.3% of the apps, respectively. *Marketing Analytics* and *Provide Ad* are the most frequent purposes found in 94.1% and 78.7% of apps'

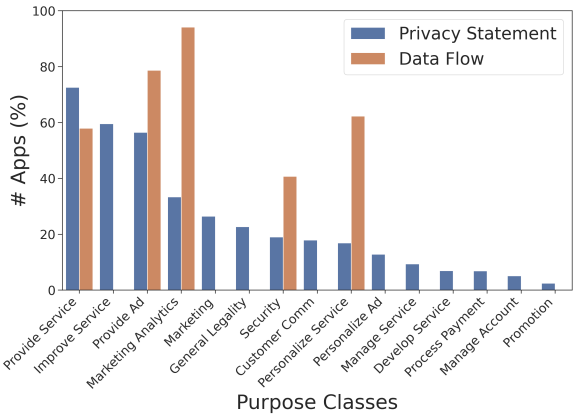

Figure 2: Distribution of purpose classes in the privacy statements and data flows of mobile apps.

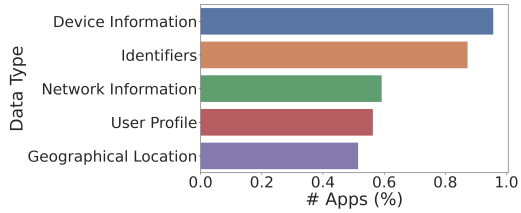

Figure 3: Distribution of data types in apps' data flows.

data flows, respectively. These results indicate that apps commonly collect both identifiable and anonymous information of devices to deliver relevant advertisements and perform data analytics. The distributions of the purposes and data types are shown in Figs. 2 and 3, respectively.

There is a mismatch between the distribution of purposes of data flows and that of privacy statements. Although the most common data-flow purposes are advertising and marketing analytics that are present in more than 78.7% of the apps, these purposes are found in privacy statements of only 56.5% and 33.4% of the apps, respectively. The significantly lower presence of the purposes in privacy policies indicates that declarations of data-usage purposes for advertising and analytics are frequently omitted in apps' policies.

8.3 End-to-end Detection of Contradictions

Evaluation Metrics. We evaluate PurPliance's end-to-end detection of contradictory sentence pairs in privacy policies. Testing the performance at the sentence level assesses the usability of the system better than at the low-level privacy statement tuples. A human analyst would need to read whole sentences to understand the context of a detected contradiction so that s/he can verify and fix it. Therefore, a low false positive rate will help human analysts reduce their effort of reviewing many non-contradictory sentences.

Dataset Creation. We create a ground-truth dataset of 108 policies selected from the privacy policy corpus (Section 8.1). To increase the diversity of the policies, we select policies of apps with different levels of popularity as popular apps may have more resources to create their policies than less popular ones. In particular, we randomly select 36 apps in each of the 3 segments based on the number of app installs: greater than 1M (3,144 apps), from 10k to 1M (10,482 apps), and less than 10k (3,176 apps). To have diverse document structures, we exclude similar policies created from templates. They are detected by a high TF-IDF cosine similarity [45] (greater than 0.95) and then manual verification that they have no significant differences other than the company/developer names. Documents that are not a valid privacy policy (e.g., terms of service or home pages), due to errors in data collection and pre-processing, are also excluded from the selection process.

Each privacy policy is independently annotated by 2 co-authors: an advanced PhD student and a researcher at a major global company, both with more than 3 years of experience in privacy research. We carefully read the policies and interpret the policy sentences as fully as possible to identify pairs of contradictory data-practice statements (detailed steps are described in Appendix K.1). Any disagreements were then resolved during follow-up discussions after every 10 policies were annotated. The annotation took two annotators 108 hours in total (30 minutes/policy/analyst on average).

There are 189 pairs of contradictory sentences in 47 (43.5%) policies. Of these policies, 32 (68.1%) contain 1–3, 12 (25.5%) contain 4–9, and 3 (6.4%) contain more than 9 sentence pairs. The dataset has 5,911 sentences where each policy has an average of 110.8 (96.4 standard deviation) sentences. The selected apps and their statistics are provided in Table 18 (Appendix K.2).

Experimental Configurations. To comparatively analyze the effects of the main components of PurPliance, we introduce the following configurations. PurPliance-PA is a *purpose-agnostic* version that does not extract purpose clauses and, thus, uses only non-purpose transformation rules T1, T3, T4 in Table 4. PurPliance-SRL is PurPliance-PA with PolicyLint's ontologies and data-practice verb list. Based on PolicyLint that uses the default parameters in its open-source repository [6], PolicyLint-PO leverages PurPliance's more complete SCoU verb list and data-object/entity ontologies.

Evaluation Results. PurPliance has 95% precision and 50% recall, which are significantly higher than 19% precision and 10% recall of PolicyLint. There are three main sources of PurPliance's improvements over PolicyLint. First, the semantic argument analysis improves the extraction of privacy statement tuples and increases both precision and recall so PurPliance-SRL improves F1 score from 20% to 32% compared to PolicyLint-PO. Second, a more complete data-practice verb list and data-object/entity ontologies improve the coverage of sentences so F1 of PurPliance-PA increases from 32% to 50% compared to PurPliance-SRL. The more complete verb list and ontologies also increase the performance of PolicyLint-PO from 13% to 20% F1 score compared to PolicyLint. Third, the analysis of data-usage purposes improves the detection of contradictions and increases the precision of PurPliance from 60% to 95% while recall is also enhanced from 43% to 50% compared to PurPliance-PA. The results are listed in Table 10.

The analysis of data-usage purposes improves PurPliance's F1 from 50% to 65% compared to PurPliance-PA. First, false positives are reduced because of the inclusion of purposes in interpreting sentences and more accurate interpretation of data-selling practices

Config	Precision	Recall	F1
PolicyLint	0.19	0.10	0.13
PolicyLint-PO	0.23	0.18	0.20
PurPliance-SRL	0.46	0.24	0.32
PurPliance-PA	0.60	0.43	0.50
PurPliance	0.95	0.50	0.65

Table 10: Detection of contradictory sentence pairs.

Config	Precision	# Statements	# Sentences
PolicyLint	0.82	85	47
PurPliance	0.91	160	68

Table 11: Performance of privacy statement extraction.

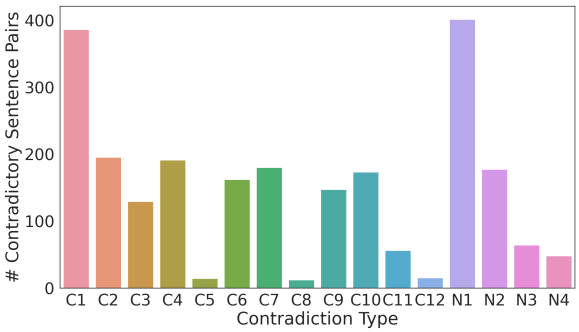

Figure 4: Distribution of potential purpose contradictions.

(e.g., *sell* and *rent*). For example, PurPliance does not flag sentences "we do not sell personal data" and "we may disclose personal data to comply with the law" because of different sharing purposes (*Marketing* vs. *Legality*), while purpose-agnostic approaches do. Second, the recall rate is enhanced because purpose-contradiction sentence pairs, such as "we use your personal data only for providing services" and "advertisers may collect personal data to deliver advertising", cannot be detected without purposes analysis.

The low precision of PolicyLint configuration is due mainly to the fundamental change of the interpretation of privacy statements. PolicyLint ignores purposes in statements and thus creates many false positives. For example, PolicyLint's interpretation of "we do not share your personal data for marketing" as "we do not share your personal data" contradicts many other data collection/sharing statements in the policy. Moreover, our metrics are more fine-grained than those used in PolicyLint [7], signifying the impact of PolicyLint's incorrect extraction. To characterize contradiction types in policies, PolicyLint [7] measured the accuracy of detecting contradictions between pairs of *sets of sentences* where sentences in a set generate the same privacy statement tuples. However, a sentence set may include both true-contradictory and false-positive ones.

The recall rate of PurPliance is still limited for three main reasons: complex sentences (29.5%), cross-sentence references (25.3%), and incompleteness of data-object ontologies (11.6%). In complex sentences, data-practice statements are often buried among other unrelated clauses (such as conditions and means of collection). The complex meaning of multiple clauses makes the separation of data-collection statements challenging. For example, "in the event of a corporate merger, your personal data is part of the transferred assets," implies the transfer of personal data without using any data-practice verb. In addition, the sentence-level analysis cannot resolve data types or entities that are defined in other sentences, such as "this information" in "we do not collect this information."

In-depth Analysis. We compare the performance of extracting privacy statement tuples, which is an important intermediate step of PurPliance and PolicyLint. As shown in Table 11, the results on 300 randomly selected sentences from the privacy policy corpus demonstrate that PurPliance significantly outperforms PolicyLint in extracting the privacy statement tuples. PurPliance has a 9% higher precision (increased from 82% to 91%), extracts 88% more privacy statements and covers 45% more sentences than PolicyLint. Note that the precision at this step is lower than the final contradictory detection because of further filtering in the later steps of contradiction analysis. The detailed experimental procedures are described in Appendix K.3.

8.4 Analysis of Policy Contradictions and Flow-to-Policy Inconsistencies

PurPliance detected 29,521 potentially contradictory sentence pairs in 3,049 (18.14%) privacy policies. Of these sentence pairs, 2,350 (7.97%) are purpose-specific, i.e., purpose-agnostic systems will miss them. For flow-to-policy inconsistencies, PurPliance detected 95,083 (13.56%) potentially inconsistent flows between the actual behavior and privacy policies in 11,399 (69.66%) of the apps with data flows extracted. Fig. 4 shows the distribution of the purpose-specific contradiction types.

The most common contradiction types are C_1 and N_1, indicating the problematic discussion of broad data-object and purpose terms in purpose-negated statements. For example, many apps state the collected personal data is not used for third parties' marketing purposes but also mention other contradicting usage purposes. The contradictions show that privacy policies frequently contain ambiguous descriptions of their data-usage purposes. Similarly, the high number of apps containing detected flow-to-policy inconsistencies indicates a prevalence of inconsistencies in mobile apps.

8.5 Findings

Finding 1. We found an issue with statements about the collection of personal data *for internal purposes only* in 28 apps, many of which have 100k-10M installs. Their policies state that "your Personal information we collected is used for internal purposes only." However, it contradicts with "we do not rent or sell your Personal information to third parties outside without your consent," because the exception clause "without your consent" indicates the sharing of personal data with third parties for third parties' purposes. On the other hand, the apps transferred a unique id and geographical location to a third-party domain with a path *client/v2/ads-service/ads*. Therefore, such data flows were inconsistent with the policies.

Finding 2. A common privacy policy template, used in 211 (0.92%) apps in our corpus, contains contradictory statements. The policy claims that their "agents and contractors may not use your personal data for their own marketing purposes." However, the policy states

later that the app employs "3rd party ad serving systems" which "allow user data to be utilized for advertising communication purposes displayed in the form of banners and other advertisements on [app name] apps, possibly based on user interests." While ad-serving systems are one of their contractors, they use the personal data for their advertising purposes (which is subsumed under marketing purposes), and user data includes user personal data, hence these statements are contradictory with respect to the purposes of marketing and advertising.

Finding 3. Apps promise that the sharing is not for marketing but later say they will. For example, a popular education app with 10M+ installs states "we do not share your personal data with third parties or corporate affiliates for their direct marketing purposes." However, the policy also states "we allow our service providers (including analytics vendors and advertising networks) to collect information about your online activities through cookies. These third parties may use this information to display advertisements on our application and elsewhere online tailored to your interests." However, displaying targeted advertisements are direct marketing and online activities (such as browsing history) that can uniquely identify a person and can thus be considered as personal data [13]. Therefore, the latter statement is contradictory to the first statement of no direct marketing purpose.

9 DISCUSSION

While PurPliance is designed to have low false positives with reasonable coverage, systematic evaluation of its recall rate is challenging because labeling privacy policies is very complex and expensive. SRL still remains a challenging task in NLP [28]. State-of-the-art SRL models [51] achieved only 87% F1 score with 85.5% recall rates. Furthermore, the SRL model used in PurPliance was trained on a generic dataset [57] and has not yet been adapted to the privacy-policy domain. Thus, its performance may be limited. However, creating a domain-adapted SRL model requires a significant effort due to the complexity of the semantic arguments [57] and large model sizes [4]; this is part of our future inquiry.

PurPliance's extraction of data flows from network traffic has two limitations. First, it cannot decode certificate-pinned traffic which, however, constitutes only $< 5\%$ of the traffic generated by top free apps [33]. Second, the input generator used in PurPliance also cannot exercise login-required apps that use external verification information. Using advanced techniques to exercise certificate-pinned and login apps will improve the coverage of an app's execution paths, thus enhancing PurPliance's recall rate. For example, recently available TextExerciser [29] can be used to generate inputs for the analysis of apps requiring a login. Although PurPliance does not capture the traffic of certificate-pinned and login-required apps, this limitation does not increase false positives, that we aim to minimize. Therefore, we leave this as our future work.

Our analysis is based on client-side information only, so it has limitations in detecting the ultimate purpose of processing on the servers. Although the analysis assumes meaningful names of app resources such as package names and URL hosts/paths, they do not always reveal the true purposes of data flows, so the extraction cannot determine purposes of certain data flows (i.e., increase false negatives). However, predicting the purposes of app behavior still provides evidence of the presence of data-usage purposes which is useful for our goal of detecting inconsistencies.Determining the exact usage purposes of data requires knowledge of server-side processing since usage information is lost once the data is received by the servers. Therefore, the detection needs to be verified by humans such as regulators and service lawyers. Since the data purpose classification has already been discussed at length and evaluated in MobiPurpose [33], developing more sophisticated and accurate data-purpose extraction is beyond the scope of PurPliance.

10 CONCLUSION

We have presented a novel analysis of data purposes in privacy policies and the actual execution of mobile apps. We have developed PurPliance, a system for automatic detection of contradictions and inconsistencies in purposes between privacy policies and apps' data transfer. Our evaluation results have shown PurPliance to significantly outperform a state-of-the-art method and detect contradictions/inconsistencies in a large number of Android apps.

ACKNOWLEDGEMENTS

The work reported in this paper was supported in part by Samsung Research and the US National Science Foundation under Grant No. CNS-1646130 and Army Research Office under Grant No. W911NF-21-1-0057.

REFERENCES

[1] 42matters AG. 2020. Google Play Categories | 42matters. Retrieved 06/27/2020 from https://42matters.com/docs/app-market-data/android/apps/google-play-categories.
[2] AdGuard. 2021. AdGuard ad filters | AdGuard knowledgebase. Retrieved 03/12/2021 from https://kb.adguard.com/en/general/adguard-ad-filters.
[3] Charu C. Aggarwal and ChengXiang Zhai. 2012. A survey of text clustering algorithms. In *Mining Text Data*. Charu C. Aggarwal and ChengXiang Zhai, editors. Springer US, Boston, MA, 77–128. DOI: 10.1007/978-1-4614-3223-4_4.
[4] AllenAI. 2020. AllenNLP - Semantic Role Labeling. (2020).
[5] Ben Andow. 2020. HtmlToPlaintext. Retrieved 07/24/2020 from https://github.com/benandow/HtmlToPlaintext.
[6] Ben Andow. 2020. PrivacyPolicyAnalysis. Retrieved 04/10/2021 from https://github.com/benandow/PrivacyPolicyAnalysis.
[7] Benjamin Andow, Samin Yaseer Mahmud, Wenyu Wang, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Tao Xie. 2019. PolicyLint: Investigating Internal Privacy Policy Contradictions on Google Play. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 585–602.
[8] Benjamin Andow, Samin Yaseer Mahmud, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Serge Egelman. 2020. Actions speak louder than words: entity-sensitive privacy policy and data flow analysis with policheck. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 985–1002.
[9] AppCensus, Inc. 2020. AppCensus AppSearch. (2020).
[10] Hannah Bast and Elmar Haussmann. 2013. Open Information Extraction via Contextual Sentence Decomposition. In

2013 IEEE Seventh International Conference on Semantic Computing, 154–159. DOI: 10.1109/ICSC.2013.36.

[11] Jaspreet Bhatia and Travis D. Breaux. 2017. A data purpose case study of privacy policies. *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 394–399.

[12] Jaspreet Bhatia and Travis D. Breaux. 2018. Semantic Incompleteness in Privacy Policy Goals. In *26th IEEE International Requirements Engineering Conference (RE'18)*.

[13] Sarah Bird, Ilana Segall, and Martin Lopatka. 2020. Replication: why we still can't browse in peace: on the uniqueness and reidentifiability of web browsing histories. In *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*. USENIX Association, 489–503.

[14] Claire Bonial, Julia Bonn, Kathryn Conger, Jena Hwang, Martha Palmer, and Nicholas Reese. 2015. *English PropBank Annotation Guidelines*.

[15] Duc Bui, Kang G. Shin, Jongmin Choi, and Jun Bum Shin. 2021. Automated extraction and presentation of data practices in privacy policies. *Proceedings on Privacy Enhancing Technologies*, 2021, 2.

[16] United States Federal Trade Commission. 1998. *Privacy online: a report to Congress*. The Commission.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. DOI: 10.18653/v1/N19-1423.

[18] Duc Bui. 2021. PurPliance. Retrieved 09/10/2021 from https://github.com/ducalpha/PurPlianceOpenSource.

[19] Elmar Haussmann. 2011. *Contextual Sentence Decomposition*. Master's thesis. University of Freiburg.

[20] Explosion AI. 2020. Models & Languages · spaCy Usage Documentation. (2020).

[21] Hossein Falaki, Dimitrios Lymberopoulos, Ratul Mahajan, Srikanth Kandula, and Deborah Estrin. 2010. A first look at traffic on smartphones. In *Proceedings of the 10th annual conference on Internet measurement - IMC '10*. ACM Press, Melbourne, Australia, 281. DOI: 10.1145/1879141.1879176.

[22] Google Chrome DevTools Team. 2020. Puppeteer Tools for Web Developers. Retrieved 02/05/2020 from https://pptr.dev/.

[23] Alessandra Gorla, Ilaria Tavecchia, Florian Gross, and Andreas Zeller. 2014. Checking app behavior against app descriptions. In *Proceedings of the 36th International Conference on Software Engineering* (ICSE 2014). Association for Computing Machinery, Hyderabad, India, 1025–1035. DOI: 10.1145/2568225.2568276.

[24] GuoShi. 2020. HttpCanary — HTTP Sniffer/Capture/Analysis. Retrieved 06/12/2020 from https://play.google.com/store/apps/details?id=com.guoshi.httpcanary&hl=en_US.

[25] Catherine Han, Irwin Reyes, Amit Elazari Bar On, Joel Reardon, Alvaro Feal, Serge Egelman, and Narseo Vallina-Rodriguez. 2019. Do You Get What You Pay For? Comparing the Privacy Behaviors of Free vs. Paid Apps. In *Workshop on Technology and Consumer Protection (ConPro 2019)*. San Francisco, CA, USA.

[26] Abram Handler, Matthew Denny, Hanna Wallach, and Brendan O'Connor. 2016. Bag of What? Simple Noun Phrase Extraction for Text Analysis. In *Proceedings of the First Workshop on NLP and Computational Social Science*. Association for Computational Linguistics, Austin, Texas, 114–124. DOI: 10.18653/v1/W16-5615.

[27] Hamza Harkous, Kassem Fawaz, Rémi Lebret, Florian Schaub, Kang G. Shin, and Karl Aberer. 2018. Polisis: Automated Analysis and Presentation of Privacy Policies Using Deep Learning. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, Baltimore, MD, 531–548.

[28] Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep Semantic Role Labeling: What Works and What's Next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, 473–483. DOI: 10.18653/v1/P17-1044.

[29] Yuyu He, Lei Zhang, Zhemin Yang, Yinzhi Cao, Keke Lian, Shuai Li, Wei Yang, Zhibo Zhang, Min Yang, Yuan Zhang, and Haixin Duan. 2020. TextExerciser: feedback-driven text input exercising for android applications. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 1071–1087. DOI: 10.1109/SP40000.2020.00071.

[30] Jeff Huang, Patrick O'Neil Meredith, and Grigore Rosu. 2014. Maximal sound predictive race detection with control flow abstraction. *ACM SIGPLAN Notices*, 49, 6, 337–348. DOI: 10.1145/2666356.2594315.

[31] Ilya Sergey. 2019. What does it mean for a program analysis to be sound? SIGPLAN Blog. Retrieved 10/03/2020 from https://blog.sigplan.org/2019/08/07/what-does-it-mean-for-a-program-analysis-to-be-sound/.

[32] Grant Jenks. 2020. Grantjenks/python-wordsegment. (2020).

[33] Haojian Jin, Minyi Liu, Kevan Dodhia, Yuanchun Li, Gaurav Srivastava, Matthew Fredrikson, Yuvraj Agarwal, and Jason I. Hong. 2018. Why Are They Collecting My Data?: Inferring the Purposes of Network Traffic in Mobile Apps. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2, 4, 173. DOI: 10.1145/3287051.

[34] Judith K. Jones. 1991. *Purpose Clauses: Syntax, Thematics, and Semantics of English Purpose Constructions. Studies in Linguistics and Philosophy*. Springer Netherlands. DOI: 10.1007/978-94-011-3478-1.

[35] Daniel Jurafsky and James H. Martin. 2019. *Speech and Language Processing, 3rd edition*. (Third Edition draft edition).

[36] Kaggle. 2020. Data leakage. Retrieved 10/16/2020 from https://kaggle.com/alexisbcook/data-leakage.

[37] Chisato Kitagawa. 1974. Purpose expressions in english. *Lingua*, 34, 1, 31–46. DOI: 10.1016/0024-3841(74)90075-8.

[38] Philip Kotler and Gary Armstrong. 2017. *Principles of marketing*. Pearson education.

[39] Daniil Larionov, Artem Shelmanov, Elena Chistova, and Ivan Smirnov. 2019. Semantic Role Labeling with Pretrained Language Models for Known and Unknown Predicates. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*. INCOMA Ltd.,

Varna, Bulgaria, 619–628. DOI: 10.26615/978-954-452-056-4_073.

[40] Yuanchun Li, Ziyue Yang, Yao Guo, and Xiangqun Chen. 2017. DroidBot: a lightweight UI-Guided test input generator for android. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, 23–26. DOI: 10.1109/ICSE-C.2017.8.

[41] Yuanchun Li, Ziyue Yang, Yao Guo, and Xiangqun Chen. 2019. Humanoid: A Deep Learning-Based Approach to Automated Black-box Android App Testing. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 1070–1073. DOI: 10.1109/ASE.2019.00104.

[42] Jialiu Lin, Shahriyar Amini, Jason I. Hong, Norman Sadeh, Janne Lindqvist, and Joy Zhang. 2012. Expectation and Purpose: Understanding Users' Mental Models of Mobile App Privacy Through Crowdsourcing. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing* (UbiComp '12). ACM, New York, NY, USA, 501–510. DOI: 10.1145/2370216.2370290.

[43] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9, Nov, 2579–2605.

[44] Christopher Manning and Hinrich Schutze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press. 657 pages.

[45] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

[46] Thor May. 1990. Purposive constructions in english. *Australian Journal of Linguistics*, 10, 1, 1–40. DOI: 10.1080/07268609008599430.

[47] Steve McConnell. 2004. *Code Complete, Second Edition*. Microsoft Press, USA.

[48] Helen Nissenbaum. 2004. Privacy as Contextual Integrity. *Washington Law Review*, 79, 39.

[49] Office of the Australian Information Commissioner. 2021. What is personal information? OAIC. Retrieved 04/21/2021 from https://www.oaic.gov.au/privacy/guidance-and-advice/what-is-personal-information/.

[50] Ehimare Okoyomon, Nikita Samarin, Primal Wijesekera, Amit Elazari Bar On, Narseo Vallina-Rodriguez, Irwin Reyes, Álvaro Feal, and Serge Egelman. 2019. On the ridiculousness of notice and consent: contradictions in app privacy policies. In *Workshop on Technology and Consumer Protection (ConPro)*. Workshop on Technology and Consumer Protection (ConPro 2019), in conjunction with the 39th IEEE Symposium on Security and Privacy. San Francisco, CA, USA.

[51] Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2018. A Span Selection Model for Semantic Role Labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 1630–1642. DOI: 10.18653/v1/D18-1191.

[52] Xiang Pan, Yinzhi Cao, Xuechao Du, Boyuan He, Gan Fang, and Yan Chen. 2018. FlowCog: context-aware semantics extraction and analysis of information flow leaks in android apps. In *Proc. USENIX Security Symposium* (SEC'18). USENIX Association, Baltimore, MD, USA, 1669–1685.

[53] Rahul Pandita, Xusheng Xiao, Wei Yang, William Enck, and Tao Xie. 2013. WHYPER: towards automating risk assessment of mobile applications. In *Proc. USENIX Security Symposium*. Presented as part of the 22nd {USENIX} Security Symposium ({USENIX} Security 13), 527–542.

[54] Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards Robust Linguistic Analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Sofia, Bulgaria, 143–152.

[55] Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*. Association for Computational Linguistics, 1–40.

[56] Zhengyang Qu, Vaibhav Rastogi, Xinyi Zhang, Yan Chen, Tiantian Zhu, and Zhong Chen. 2014. AutoCog: Measuring the Description-to-permission Fidelity in Android Applications. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (CCS '14). ACM, New York, NY, USA, 1354–1365. DOI: 10.1145/2660267.2660287.

[57] Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2013. OntoNotes Release 5.0 LDC2013T19. Linguistic Data Consortium. (2013).

[58] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. EMNLP-IJCNLP 2019. Association for Computational Linguistics, Hong Kong, China, 3982–3992. DOI: 10.18653/v1/D19-1410.

[59] Irwin Reyes, Primal Wiesekera, Abbas Razaghpanah, Joel Reardon, Narseo Vallina-Rodriguez, Serge Egelman, and Christian Kreibich. 2017. "is our children's apps learning?" automatically detecting COPPA violations. In *Workshop on Technology and Consumer Protection (ConPro 2017)*. Workshop on Technology and Consumer Protection (ConPro 2017), in conjunction with the 38th IEEE Symposium on Security and Privacy (IEEE S&P 2017). San Jose, CA, USA.

[60] Irwin Reyes, Primal Wijesekera, Joel Reardon, Amit Elazari Bar On, Abbas Razaghpanah, Narseo Vallina-Rodriguez, and Serge Egelman. 2018. "Won't Somebody Think of the Children?" Examining COPPA Compliance at Scale. *Proceedings on Privacy Enhancing Technologies*, 2018, 3, 63–83. DOI: 10.1515/popets-2018-0021.

[61] Lior Rokach and Oded Maimon. 2005. Clustering methods. In *Data mining and knowledge discovery handbook*. Springer, 321–352.

[62] Peng Shi and Jimmy Lin. 2019. Simple BERT Models for Relation Extraction and Semantic Role Labeling. *arXiv:1904.05255 [cs]*.

[63] Anastasia Shuba and Athina Markopoulou. 2020. NoMoATS: towards automatic detection of mobile tracking. *Proceedings on Privacy Enhancing Technologies*, 2020, 2, 45–66. DOI: 10.2478/popets-2020-0017.

[64] Yan Shvartzshnaider, Noah Apthorpe, Nick Feamster, and Helen Nissenbaum. 2019. Going against the (Appropriate) Flow: A Contextual Integrity Approach to Privacy Policy Analysis. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, 7, 1, 162–170.

[65] Rocky Slavin, Xiaoyin Wang, Mitra Bokaei Hosseini, James Hester, Ram Krishnan, Jaspreet Bhatia, Travis D. Breaux, and Jianwei Niu. 2016. Toward a framework for detecting privacy policy violations in android application code. In *Proceedings of the 38th International Conference on Software Engineering* (ICSE '16). ACM, New York, NY, USA, 25–36.

[66] Yannis Smaragdakis, Jacob Evans, Caitlin Sadowski, Jaeheon Yi, and Cormac Flanagan. 2012. Sound predictive race detection in polynomial time. In *Proceedings of the 39th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages* (POPL '12). Association for Computing Machinery, New York, NY, USA, 387–400. DOI: 10.1145/2103656.2103702.

[67] Gaurav Srivastava, Kunal Bhuwalka, Swarup Kumar Sahoo, Saksham Chitkara, Kevin Ku, Matt Fredrikson, Jason Hong, and Yuvraj Agarwal. 2018. PrivacyProxy: Leveraging Crowdsourcing and In Situ Traffic Analysis to Detect and Mitigate Information Leakage. *arXiv:1708.06384 [cs]*.

[68] TermsFeed. 2021. Personal vs. sensitive information. TermsFeed. Retrieved 04/21/2021 from https://www.termsfeed.com/blog/personal-vs-sensitive-information/.

[69] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-Read Students Learn Better: On the Importance of Pre-training Compact Models. *arXiv:1908.08962 [cs]*.

[70] X. Wang, X. Qin, M. Bokaei Hosseini, R. Slavin, T. D. Breaux, and J. Niu. 2018. GUILeak: Tracing Privacy Policy Claims on User Input Data for Android Applications. In *40th International Conference on Software Engineering (ICSE)*, 37–47. DOI: 10.1145/3180155.3180196.

[71] Shomir Wilson, Florian Schaub, Aswarth Abhilash Dara, Frederick Liu, Sushain Cherivirala, Pedro Giovanni Leon, Mads Schaarup Andersen, Sebastian Zimmeck, Kanthashree Mysore Sathyendra, N. Cameron Russell, Thomas B. Norton, Eduard Hovy, Joel Reidenberg, and Norman Sadeh. 2016. The creation and analysis of a website privacy policy corpus. In *Proc. ACL 2016*. Association for Computational Linguistics, 1330–1340. DOI: 10.18653/v1/P16-1126.

[72] Le Yu, Xiapu Luo, Xule Liu, and Tao Zhang. 2016. Can We Trust the Privacy Policies of Android Apps? In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 538–549. DOI: 10.1109/DSN.2016.55.

[73] Le Yu, Xiapu Luo, Xule Liu, and Tao Zhang. 2016. Can We Trust the Privacy Policies of Android Apps? In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 538–549. DOI: 10.1109/DSN.2016.55.

[74] Sebastian Zimmeck and Steven M. Bellovin. 2014. Privee: an architecture for automatically analyzing web privacy policies. In *23rd USENIX Security Symposium*. 23rd {USENIX} Security Symposium, 1–16.

[75] Sebastian Zimmeck, Ziqi Wang, Lieyong Zou, Roger Iyengar, Bin Liu, Florian Schaub, Shomir Wilson, Norman Sadeh, Steven M. Bellovin, and Joel Reidenberg. 2017. Automated Analysis of Privacy Requirements for Mobile Apps. In *Proceedings 2017 Network and Distributed System Security Symposium*. Internet Society, San Diego, CA. DOI: 10.14722/ndss.2017.23034.

A SEMANTIC ARGUMENTS OF PURPOSE CLAUSES

Semantic arguments of an event do not change even though the syntactic structure of the sentence changes. For example, let us consider the following sentences which express a data-usage event:

- $[\text{We}]_{\text{Arg0}}$ do not $[\text{share}]_{\text{V}}$ $[\text{your personal data}]_{\text{Arg1}}$ $[\text{with third parties}]_{\text{Arg2}}$ $[\text{for targeted ads}]_{\text{Argm-Pnc}}$;
- $[\text{Third parties}]_{\text{Arg0}}$ may not $[\text{collect}]_{\text{V}}$ $[\text{your personal data}]_{\text{Arg1}}$ $[\text{to deliver targeted ads}]_{\text{Argm-Pnc}}$.

While the purpose of *delivering targeted ads* is stated differently in noun and verb phrases starting with *for* and *to*, it is consistently an *Argm-Pnc* (purpose-not-cause) argument of the predicate. The data object *your personal data* is also an *Arg1* in both cases.

Table 12 lists the predicate-specific semantic arguments of purpose clauses used in addition to the common arguments *Argm-Prp* and *Argm-Pnc*.

Predicates	Argument
use, save, check	Arg2
analyze	Argm-Adv
save, receive, solicit, record	Arg3
receive	Arg4
disclose, give, sell, send, transmit, provide	C-Arg1

Table 12: Predicate-specific semantic arguments of purpose clauses used by `PurPliance`.

B EXAMPLES OF PREDICATE-OBJECT PAIRS

Table 13 shows examples of purpose classification with PO pairs.

Purpose clause	PO pairs
To provide personalized services	(provide, personalized services), (personalize, services)
To comply with laws	(comply, laws)
For promotional purposes	(, promotional purposes)
For scientific purposes	(, scientific purposes)

Table 13: Examples of purpose classification with PO pairs.

C POLICY PURPOSE PREDICTION PERFORMANCE

The performance of policy purpose prediction is shown in Table 14.

High-level	Low-level	Precision
Production	Develop service	100.0
	Improve service	100.0
	Manage account	100.0
	Manage service	100.0
	Personalize service	83.3
	Process payment	100.0
	Provide service	100.0
	Security	100.0
Marketing	Customer comm.	80.0
	General marketing	100.0
	Marketing analytics	100.0
	Personalize ad	100.0
	Provide ad	100.0
	Promotion	100.0
Legality	General legality	100.0
Other	Other purposes	100.0
	Average	97.8

Table 14: Policy purpose prediction performance on test set.

D PURPOSE APPROXIMATION PROOF

The following is the proof of Theorem 6.6.

Proof. (1) Because $q_i \approx_\kappa q_j$, exists q' such as $q' \sqsubset_\kappa q_i$ and $q' \sqsubset_\kappa q_j$. Therefore, $(e_i, q') \sqsubset_\pi (e_i, q_i)$ and $(e_i, q') \sqsubset_\pi (e_j, q_j)$. The existence of $p' = (e_i, q')$ implies $p_i = (e_i, q_i) \approx_\pi p_j = (e_j, q_j)$.

(2) Because $q_i \approx_\kappa q_j$, exists q' such as $q' \sqsubset_\kappa q_i$ and $q' \sqsubset_\kappa q_j$. Therefore, $(e_i, q') \sqsubset_\pi (e_i, q_i)$. Also, because given $e_i \sqsubset_\epsilon e_j$, so $(e_i, q') \sqsubset_\pi (e_j, q_j)$. The existence of $p' = (e_i, q')$ implies $p_i = (e_i, q_i) \approx_\pi p_j = (e_j, q_j)$.

(3) The proof is similar to (2) with the roles of entities e_i and e_j swapped with purposes q_i and q_j, respectively.

(4) The proof is similar to (3) with the roles of entities e_i and e_j swapped with purposes q_i and q_j, respectively.

(5) Because $e_i \approx_\kappa e_j$, exists e' such as $e' \sqsubset_\epsilon e_i$ and $e' \sqsubset_\kappa e_j$. Because $q_i \approx_\kappa q_j$, exists q' such as $q' \sqsubset_\kappa q_i$ and $q' \sqsubset_\kappa q_j$. Therefore, $(e', q') \sqsubset_\pi (e_i, q_i)$ and $(e', q') \sqsubset_\pi (e_j, q_j)$. The existence of $p' = (e', q')$ implies $p_i = (e_i, q_i) \approx_\pi p_j = (e_j, q_j)$.

□

E DATA FLOW PURPOSE FEATURES

Features used for inferring usage purposes of data flows are listed in Table 15. The ablation study results are shown in Table 16.

F PRIVACY POLICY CRAWLER AND PREPROCESSOR

A crawler was developed to scrap the privacy policies of Android apps. Given an app ID, the crawler first searches for the privacy policy URL in the metadata of the app on Google Play Store. A full HTML version of the web page is scrapped by using Google Chrome controlled by Puppeteer web driver [22] so that dynamically rendered privacy notice contents are downloaded correctly. Finally, PolicyLint's open-source privacy policy HTML pre-processing tool [5] was used to remove extraneous GUI elements and HTML tags and extract a plain-text version that contains well-formed sentences of the privacy policy. If the privacy policy classifier determines that the downloaded document is not a privacy policy, the crawler searches for a privacy link within the page and repeats the HTML downloading and extraction process.

A classifier based on Support Vector Machine (SVM) is developed to determine whether the downloaded web document is a privacy policy or not. The model is trained on a set of 375 documents (199 positive and 176 negative examples). The training and validation used 5-fold cross validation while 15% of the documents were held out for testing. The classifier achieved F1 scores of 98.12% and 96.49% for validation and testing, respectively. Similar to PolicyLint [7], we filtered out sentences that do not contain any data practice verbs or data objects, and sentences that start with an interrogative word.

G MAPPING PURPOSES OF MOBIPURPOSE TO PURPLIANCE'S PURPOSE TAXONOMY

The conversion from purpose categories in MobiPurpose to the data-usage taxonomy of `PurPliance` is listed in Table 17.

H DOMAIN-ADAPTED NER MODEL

`PurPliance` uses a domain-adapted NER model to extract the data objects and entities from sentences. We retrained the NER component of the Spacy *en_web_core_lg* language model [20] on PolicyLint's dataset of 600 manually annotated sentences. 150 sentences were randomly selected while the other 450 have one of the 9 subsumptive relationship patterns. Similar to the procedure used in PolicyLint [7], we trained the model on the training set of 500 samples until the loss converges after 180 epochs. The data object recognition performance on the test set of 100 samples achieves an 83.1% F1 score (82.26% precision and 83.95% recall).

I DISTRIBUTION OF APPS AND POLICIES

Fig. 5 shows the distribution of apps and unique policies per app category.

J DISTRIBUTION OF CAPTURED TRAFFIC OVER APP CATEGORIES

Statistics of network traffic intercepted are shown in Fig. 6. The top 3 contacted domains are *googleads.g.doubleclick.net* with 230,309 requests, *pagead2.googlesyndication.com* with 86,767 requests, and *csi.gstatic.com* with 73,939 requests. The traffic distribution has a long tail: 13,269 (68.8%) domains were contacted by only one app and 12,561 (65.1%) domains have less than 10 network data requests.

Group	Feature	Explanation	Dimension
Sent data	(G1) URL bag-of-words	Bag of words extracted from the request URL.	140
	(G2) Sent data bag-of-words	Bag of words extracted from the sent HTTP(S) data.	140
Data characteristics	(G3) Sent data types	Enumeration of data types in the sent data.	6
	(G4) Number of key-values	Number of key-value pairs in the sent data.	1
	(G5) Number of data types	Number of data types in the sent data.	1
App-specific info	(G6) App-destination similarity	The package name has long common substrings with the URL.	3

Table 15: Features used in the purpose classification for data flows.

Features	Precision	Recall	F1
G.1	0.69	0.67	0.68
G.1,2	0.73	0.68	0.70
G.1,2,3	0.75	0.73	0.74
G.1,2,3,4	0.77	0.75	0.75
G.1,2,3,4,5	0.79	0.76	0.77
G.1,2,3,4,5,6	0.81	0.78	0.79

Table 16: Ablation study of the purpose classification features. The performance is on the test set.

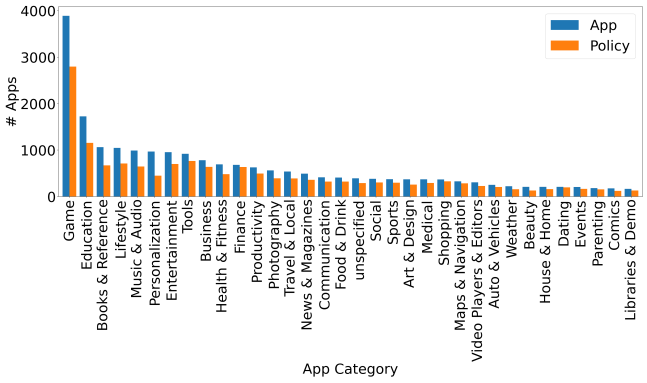

Figure 5: Distribution of apps and unique policies per app category.

K DATASET FOR END-TO-END CONTRADICTION DETECTION

K.1 Annotation Procedure

Contradictions in each privacy policy are identified as follows. We first look for any sentences that contain negated sentiment either in data collection/sharing or in purpose clauses as their occurrences are much less frequent than positive ones. For each negated sentence found, we try to find as many contradictory positive statements as possible. The common keywords in negated statements include "not", "never", "only for", "only to", "solely". However, because negated statements are expressed in various ways and their meanings can only be determined by the context, we need to read the whole policies to search for negated statements. To fully interpret the policy sentences, we checked the meaning of each word and the contextual sentences of each identified statement to understand the specific intention and meaning of the terms in the sentence. We also consulted external regulatory texts for the definition of certain data types when necessary.

K.2 Dataset

The apps selected for the evaluation of end-to-end contradiction detection and their statistics are shown in Table 18. The app with the most contradictory sentence pairs is *au.com.realestate.app*. It contains a statement that "We do not collect sensitive information as defined under the Privacy Act 1988("Privacy Act")." However, because *sensitive information* is a type of *personal information* as defined by the Privacy Act [49, 68], the broad negated-sentiment statement have a narrowing-definition contradiction with many other sentences about collection/sharing of *personal information*.

K.3 Evaluation of Privacy Statement Extraction

Experimental Procedure. We compare `PurPliance`'s performance in extracting privacy statements from policy document sentences with PolicyLint [7], a state-of-the-art extraction method. To avoid test data leakage [36], 285k (20%) sentences in the corpus were set aside as the test set while `PurPliance` was developed and fine-tuned on the other 80% sentences. 300 sentences were then randomly selected from the test set for evaluation. The privacy statements of `PurPliance` and PolicyLint from their parameter extraction step are used. We used PolicyLint's public implementation without any changes. The NER models used by both systems are trained on the same dataset, and hence they have similar capabilities. In addition, since PolicyLint does not support purpose extraction, purposes extracted by `PurPliance` and their combinations are not counted in this evaluation. Three of the authors annotated the sentences. We used majority votes and held discussions to reach a consensus about the correctness of the extracted statements.

Metrics. Since our goal is to minimize false positives, the precision and the number of extracted statements are used as the main performance metrics. Different from creating a dataset of contradictory sentences in Section 8.3, there are a large number of possible text spans that express a data type or a receiving entity in each sentence and limitations of the contiguous entity annotation. Therefore, it requires a significant amount of effort to create a complete dataset of annotations of all policy statements and control its quality [15].

Results and Analysis. Our results show that `PurPliance` extracts more privacy statements with higher precision than PolicyLint. The precision of `PurPliance` is 0.91, higher than 0.82 of PolicyLint.

	MobiPurpose purpose class	PurPliance purpose class
1	Search nearby places	Production - Provide service
2	Geosocial networking	Production - Provide service
3	Network switch notification	Production - Provide service
4	Geotagging	Production - Provide service
5	Transportation information	Production - Provide service
6	Map and navigation	Production - Provide service
7	Recording	Production - Provide service
8	Location-based game	Production - Provide service
9	Alert and remind	Production - Provide service
10	Third-party login	Production - Provide service
11	Geo localization	Production - Provide service
12	Reverse geocoding	Production - Provide service
13	Location spoofing	Production - Provide service
14	Network optimization	Production - Provide service
15	Interface customization	Production - Personalize service
16	Location-based customization	Production - Personalize service
17	Signed-out user personalization	Production - Personalize service
18	Anti-fraud	Production - Security
19	Authentication	Production - Security
20	User/device tracking for data analytics	Marketing - Marketing analytics
21	Data collection for analytics	Marketing - Marketing analytics
22	Data collection for advertising	Marketing - Provide ad
23	User/device tracking for advertising	Marketing - Provide ad
24	Data collection for advertising personalization	Marketing - Personalize ad

Table 17: Conversion from purpose classes in MobiPurpose [33] to PurPliance taxonomy. This table does not present full PurPliance taxonomy but relevant classes with ones in MobiPurpose.

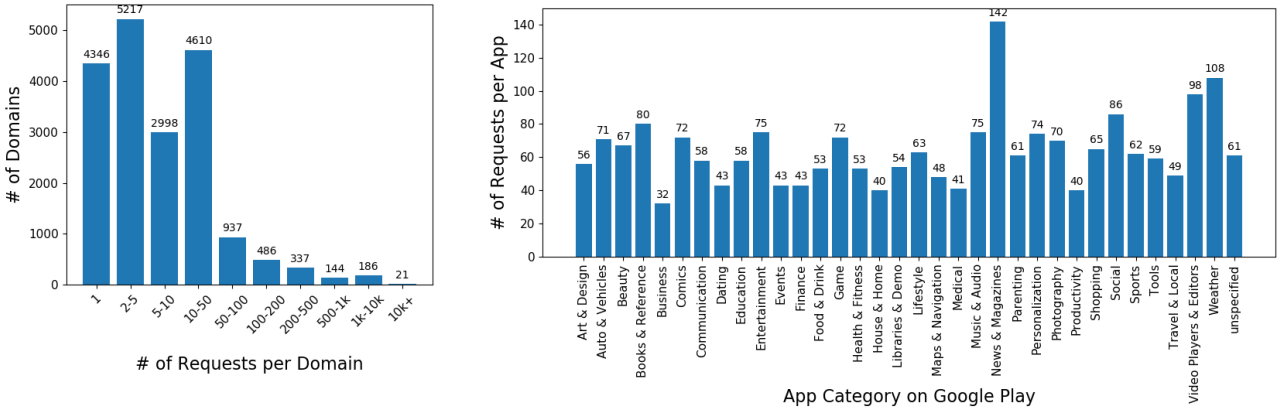

Figure 6: Data statistics of 1,727,001 network requests intercepted. The left figure shows the distribution of requests among domains. The right figure shows the distribution of requests among app categories on Google Play.

PurPliance extracted 160 statements from 68 sentences which are 88% more statements and cover 45% more sentences than PolicyLint. Table 11 shows our experimental results.

An in-depth analysis shows the most common incorrect extraction of both systems is caused by the erroneous recognition of data objects and receivers by NER models. Furthermore, since both systems do not analyze the semantics of sentences, they extract data-collection practices from non-data-collection statements such as "data protection laws in Europe distinguish between organizations that process personal data ..." However, both systems employ further filtering in the later steps of their pipelines so trivial incorrectness would not increase false positive rates of the whole system significantly.

PurPliance extracts more statements than PolicyLint because it can cover many grammar variations which are not included in PolicyLint's 16 sentence templates of data collection and sharing. For example, PolicyLint missed all policy statements from "we do not sell, trade, or otherwise transfer to outside parties your personal identifiable information," because it did not recognize the long list of multiple data action verbs.

	App	# Sent-Pairs	# Sentences	# Installs
1	au.com.realestate.app	31	264	1,000,000
2	com.rfi.sams.android	18	468	10,000,000
3	com.toongoggles.tv	13	137	100,000
4	in.followon.alumni	9	143	100
5	com.birthday.flowers.images	8	71	1,000
6	com.SuperAwesome.DragonVillageBlast	7	213	100,000
7	com.qarasoft.kosho	7	121	50,000
8	com.crazyplex.hotcoffeemaker	6	148	100,000
9	com.innovle.qtix	6	86	5,000
10	com.colorflash.callerscreen	5	77	1,000,000
11	com.mobibah.afanoromolovesms	5	35	10,000
12	com.theepochtimes.news	4	145	100,000
13	net.playtouch.becomeapuppygroomer	4	122	10,000
14	com.spicyyoghurt.pixiegame.free	4	37	100
15	com.appwallet.magictoucheffect	4	32	10,000
16	com.squareup	3	474	10,000,000
17	com.tappx.flipnsave.battery	3	285	1,000,000
18	com.greatclips.android	3	280	5,000,000
19	com.fishcrackergames.WhatBread	3	52	500
20	com.fontskeyboard.fonts	3	46	5,000,000
21	com.qvq.simpleball	3	45	500
22	com.grab.yourbaby	3	43	5,000
23	com.pdffilereader	3	36	1,000,000
24	com.visionsmarts.pic2shop	3	19	1,000,000
25	com.gi.talkingrapper	2	365	1,000,000
26	com.olo.kneaders	2	224	10,000
27	com.geeko.ivrose	2	163	1,000,000
28	com.ilsc.mygreystone	2	156	100
29	me.nextplus.smsfreetext.phonecalls	2	137	5,000,000
30	com.eivaagames.Bowling3DPro	2	79	1,000,000
31	com.dumpgames.virtual.single.dad.simulator.happy.father	2	75	500,000
32	theme.space.galaxy.planet.shining.aircraft.launcher.wallpaper	2	36	100
33	comethru.event.organizer	1	281	10
34	com.bravolang.chinese	1	272	1,000,000
35	com.sia.id00145	1	173	100
36	com.journedelafemme.bestwomanslove	1	89	1,000
37	com.lily.times.basset2.all	1	80	1,000,000
38	com.appybuilder.bmkbmk767.purerelationship	1	71	100
39	com.lwsipl.archightech.launcher	1	66	500,000
40	com.lexilize.notme	1	58	500
41	com.polaroid.cube.plus	1	56	1,000
42	kynguyen.app.mirror	1	43	1,000,000
43	com.repsi.heartrate	1	35	1,000,000
44	appinventor.ai_mssrnick.almohana	1	27	100
45	air.com.miracle.SeaRescue	1	24	500
46	photo.editor.collage.maker.photoeditor	1	12	1,000,000
47	net.moderndefense	1	10	10,000
	Total	189	5911	

Table 18: Selected apps with contradictory sentence pairs. *# Sent-Pairs* stands for the number of contradictory sentence pairs.