Disambiguating Requirements through Syntax-Driven Semantic Analysis of Information Types

Mitra Bokaei Hosseini[1], Rocky Slavin[2], Travis D. Breaux[3], Xiaoyin Wang[2], and Jianwei Niu[2]

[1] St. Mary's University, San Antonio, TX, USA mbokaeihossein@stmarytx.edu
[2] University of Texas at San Antonio, San Antonio, TX, USA
{rocky.slavin,xiaoyin.wang,jianwei.niu}@utsa.edu
[3] Carnegie Mellon University, Pittsburgh, PA, USA tdbreaux@andrew.cmu.edu

Abstract. [context and motivation] Several state laws and app markets, such as Google Play, require the disclosure of app data practices to users. These data practices constitute critical privacy requirements statements, since they underpin the app's functionality while describing how various personal information types are collected, used, and with whom they are shared. [Question/Problem] When such statements contain abstract terminology referring to information types (e.g., "we collect your device information"), the statements can become ambiguous and thus reduce shared understanding among app developers, policy writers and users. [Principle Ideas/Results] To overcome this obstacle, we propose a syntax-driven method to infer semantic relations from a given information type. We use the inferred relations from a set of information types (i.e. lexicon) to populate a partial ontology. The ontology is a knowledge graph that can be used to guide requirements authors in the selection of the most appropriate information type terms. [Contributions] Our method employs a shallow typology to categorize individual words in an information type, which are then used to discharge production rules in a context-free grammar (CFG). The CFG is augmented with semantic attachments that are used to generate the semantic relations. This method is evaluated on 1,853 unique information types from 30 privacy policies to yield 0.99 precision and 0.91 recall when compared to human interpretation of the same information types.

Keywords: Privacy policy · Abstraction · Ontology.

1 Introduction

Mobile and web application (app) companies manage data practice requirements concerning information collection, use, and sharing. These requirements are communicated to users through privacy policies [1,18]. When describing data practices, privacy policies often use vague, high-level terms with unclear conditions to generalize a wide range of information types [29]. To be comprehensive, the language used in these policies tends to be ambiguous, which consequently leads to multiple, unwanted interpretations [25]. Ambiguity can also reduce the shared understanding among app developers, policy writers, and regulators who need to support privacy compliance and data transparency [7]. Such misunderstanding has consequences, such as the recent $5 billion

settlement of Federal Trade Commission with Facebook [16]. This penalty arose from poor data practices resulting in leaking the personal information of 87 million users to third parties.

To ensure data transparency and compliance, methods have been proposed to analyze data practices in privacy policies. For example, Breaux et al. formalized data practice requirements from privacy policies using Description Logic [6], to automatically detect conflicting requirements across interacting services [8]. Tracing privacy requirements across policies can enhance developers' understanding of third-parties' data practices and comply with legal requirements, such as General Data Protection Regulation (GDPR), Articles 13.1 and 14.12. Other researchers have proposed techniques to trace requirements from privacy policies to app code using lookup tables, platform permissions, and information flow analysis [28,34]. These methods were based on a manually-compiled lexicon (i.e. set of information types), wherein information types were grouped into categories tagged by keywords, such as "location," "contact," or "identifier" [34]. Coarse categorization can lead to inaccuracies, e.g., the phrase "WiFi SSID" can be construed to be a type of location information [34], perhaps because the corresponding technology can be used to infer device locations; however, this type does not constitute a location.

Hypernymy occurs when a more abstract or general information type is used instead of a more specific information type (e.g., the broader term "device information" used in place of "mobile device identifier") [3]. Hypernymy permits multiple interpretations of words and phrases, which leads to ambiguity and inconsistency in traceability.

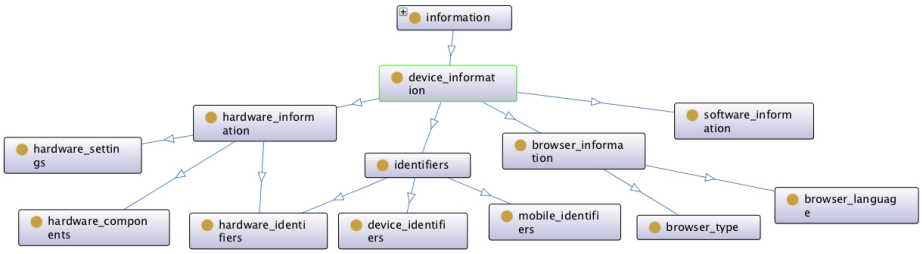

Fig. 1. Ontology Example

Consider the following snippet from EA Games' privacy policy[4] stating, "We collect other information automatically [...], including: [...]; Mobile and other hardware or device identifiers; Browser information, including your browser type and the language you prefer; [...]; Information about your device, hardware and software, such as your hardware settings and components [...]". In this example, an analyst may make several inferences: (1) that "mobile identifiers," "hardware identifiers," and "device identifiers" are all kinds of "identifiers" that EA collects; (2) that "browser type" and "browser language" are both kinds of "browser information;" (3) "hardware information" and "software information" can be inferred as specific kinds of "device information;" and (4) that "hardware settings and components" are a specific kind of "hardware information." The analyst can infer such hypernymy relationships between information

[4] https://www.ea.com/legal/privacy-policy

types intuitively by applying their domain knowledge and experience. An analyst who documents these inferences could create a reusable ontology, shown in Figure 1, to illustrate each term and and it's semantic relationships to other terms via hypernymy.

Ontologies are useful in dealing with requirements that are presented in potentially abstract human language. Without an ontology, analysts may be inconsistent in their interpretations by inconsistently applying heuristics in an ad hoc manner. In contrast, ontologies enable precise, reusable and semi-automated analysis of requirements [32,33,8,9].

Prior work on ontology construction has relied on manual comparison of information types [32], which is tedious and still susceptible to human error due to fatigue and gaps in analyst domain knowledge. Furthermore, the language use evolves, requiring ontology reconstruction. Two recent studies employed regular expressions that were hand-crafted from individual policy statements to extract hypernymy [21,12]. These approaches require a new analysis for each new policy, which does not generalize well.

To summarize, the research has shown the significance of utilizing ontologies in disambiguating vague and abstract requirements [32,33,8,9]. However, the current ontology construction methods rely on manual analysis, lack scalability or validation on information types from various domains (e.g., app categories and data practices). To address these issues and enable easier, more consistent ontology construction, we propose a syntax-driven semantic analysis method to construct an ontology. This method is evaluated on information types from six domains of mobile and web-based privacy requirements considering various data practices. The contributions of this paper are two-fold: (1) a syntax-driven method to infer semantic relations from a given information type. This method is based on the principle of compositionality, which states the meaning of each phrase can be derived from the meaning of its constituents [15,23]. Using this principle, we developed a context-free grammar (CFG) augmented with semantic attachments [2] over typed constituents of an information type to infer semantic relations between the information type and its constituents. (2) an empirical evaluation of our syntax-driven semantic analysis method on sample set of 1,138 information types from 30 mobile and web-based apps' requirements in six domains, including shopping, telecommunication, social networks, employment, health, and news.

This paper is organized as follows. In § 2 and § 3, we discuss important terminology and related work. In § 4 we introduce our method. In § 5, we present the evaluation and results, followed by threats to validity and concluding remarks in § 6 and § 7.

2 Background

In this section, we introduce terminology, datasets, and research method used throughout this paper.

Hypernymy: a relationship between two noun phrases where the meaning of one (hypernym) is more generic than the other (hyponym), e.g., "device information" is a hypernym of "device ID."

Meronymy: a part-whole relationship between two noun phrases, e.g., "device ID" is a part of "device."

Synonymy: a relationship between two noun phrases with a similar meaning or abbreviation, e.g., "IP" is synonym of "Internet protocol."
Lexicon: a collection or list of noun phrases that are information type names.
Ontology: an arrangement of concept names in a graph in which terms are connected via edges corresponding to semantic relations, such as hypernymy and synonymy, among others [24]. In this paper, we only consider information type names.
Morphological Variant: a concept name that is a variant of a common lexeme, e.g., "device ID" is a morphological variant of "device."

In the definitions above, we assume that noun phrases expressed in text have a corresponding concept and that the text describes one name for the concept. This relationship between the phrase and concept is also arbitrary, as noted by Saussure in his theory of the signifier, which is the symbol that represents a meaning, and the signified, which is the concept or meaning denoted by the symbol [11]. Peirce defines a similar relationship between sign-vehicles and objects, respectively [20].
Context-free Grammar: a set of production rules, expressing the way that symbols of a language can be grouped and ordered together [24].
Semantic Attachment: each production rule in a grammar is mapped to its semantic counterpart, called semantic attachment [2].

The following are the datasets used through out this paper for the purpose of method construction and evaluation.
Lexicon L_1: a previously published lexicon containing 351 platform-related information types (e.g., "IP address") defined as "any information that the app or another party accesses through the mobile platform that is not unique to the app." The information types were extracted from collection data practices of 50 mobile app privacy policies [21,32].
Lexicon L_2: a previously published lexicon containing 1,853 information types related to any data collection, use, retention, and sharing practices, extracted from 30 mobile and web app privacy policies across six domains (shopping, telecommunication, social networks, employment, health, and news) [12].

We now describe the research method used in this paper.
Grounded Theory: a qualitative inquiry approach that involves applying codes to data through coding cycles to develop a theory grounded in the data [31]. We describe three applications [10] in this paper: (1) codes applied to phrases in Lexicon L_1 to construct a context-free grammar; (2) memo-writing to capture results from applying the grammar and its semantic attachments to infer ontological relations from L_1; and (3) theoretical sampling to test the grammar and its semantic attachments on a sample set of information types in lexicon L_2.

3 Related Work

In this section, we review related work, including how natural language affects the interpretation of requirements in requirements acquisition, documentation, and verification [30], lexicons and ontologies.

Lexicons play an important role in reducing ambiguity and improving the quality of specifications [17]. Boyd et al. proposed to reduce ambiguity in controlled natural languages by optimally constraining lexicons using term *replaceability* [5]. Our proposed

method improves lexicon development through automation to account for discovering new, previously unseen terms. By incorporating semantic relationships between terms, a lexicon can be expanded into an ontology. Breitman and do Prado Leite describe how ontologies can be used to analyze web application requirements [9]. Breaux et al. use an ontology to identify conflicting requirements across vendors in a multi-stakeholder data supply chain [8]. Their proposed ontology was formalized for three apps (i.e., Facebook, Zynga, and AOL) and contains hierarchies for actors roles, information types, and purposes. Their work motivates the use of ontologies in requirements analysis, yet relies on a small set of policies and has not been applied at scale.

Oltramari et al. propose using a formal ontology to specify privacy-related data practices [27]. The ontology is manually populated with practice categories, wherein each practice has properties, including information type. While the ontology formalizes natural language privacy requirements, there are no semantic relations formalized among information types, thus the ontology does not encode hypernymy.

Zimmeck et al. proposed an approach to identify the misalignments between data practices expressed in privacy requirements and mobile app code [34]. The approach uses a bag-of-words for three information types: "device ID", "location", and "contact information." For example, "IP address" is contained in the bag-of-words associated with device ID. Without the relationships described in an ontology, this approach cannot distinguish between persistent and non-persistent types, which afford different degrees of privacy risk to users.

Slavin et al. identify app code that is inconsistent with privacy policies using a manually constructed information type ontology [32,22]. The approach overcomes the limitation of Zimmeck et al. [34] and exemplifies the efficacy of ontologies for requirements traceability. However, it is costly and lacks scalability due to: (1) the time spent by analysts to compare information types, and (2) errors generated by analysts during comparison [22].

Hosseini et al. [21] proposed 26 regular expression patterns to parse the information types in lexicon L_1 (see § 2) and to infer semantic relations based on their syntax. The discovered patterns fail to cover all the information types in lexicon L_1 and the approach requires extending the pattern set for new policies. To address this problem, we propose a context-free grammar to formally infer all the information types in L_1 with regard to pre-defined inference heuristics that are policy-independent.

Lexical ontologies, such as WordNet, can be used in requirements analysis. WordNet is a lexical database that contains English words grouped into nouns, verbs, adjectives, adverbs, and function words [26,13]. Within each category, the words are organized by their semantic relations, including hypernymy, meronymy, and synonymy [13]. However, previous analysis shows that only 14% of information types from a privacy policy ontology [22] are found in WordNet [26], mainly because the lexicon is populated with multi-word, domain-specific phrases. Therefore, finding a category of information type along with its subordinate terms can be a challenging task for a requirement analyst. Our proposed approach aims to address this limitation in WordNet and facilitate automated analysis of data requirements.

4 Ontology Construction Method

Figure 2 presents our method overview given a privacy policy lexicon. This figure is summarized as follows: in step 1, information types in a lexicon are pre-processed and reduced; in step 2, an analyst manually assigns semantic roles to the words in each reduced information type, a step that is linear in effort in the size of the lexicon; in step 3, a context-free grammar (CFG) and its semantic attachments are used to automatically infer morphological variants and candidate ontological relations.

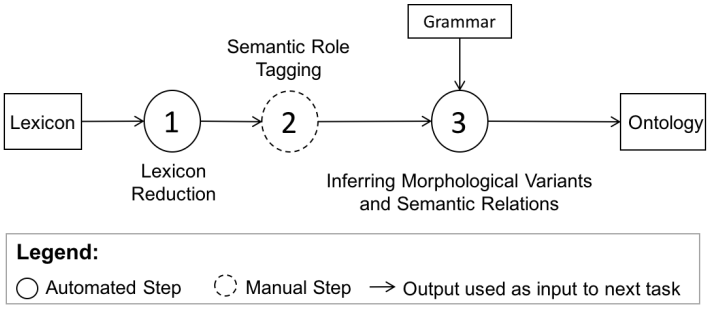

Fig. 2. Ontology Construction Method Overview

The production rules that comprise the CFG and that are introduced in this paper are used to formalize and analyze the syntax of a given information type. To infer semantic relations, we implement the rule-to-rule hypothesis [2] by mapping each production rule in the CFG to its semantic counterpart, presented using λ-calculus. We now discuss the steps in our method.

4.1 Lexicon Reduction

In step 1, the information types from the input lexicon are reduced as follows: (1) plural nouns are changed to singular nouns, e.g., "peripherals" is reduced to "peripheral;" (2) possessives are changed to non-possessive form, e.g., "device's information" is reduced to "device information;" and (3) suffixes "-related," "-based," and "-specific" are removed, e.g., "device-related" is reduced to "device."

4.2 Semantic Role Tags

Given the reduced lexicon as input, step 2 consists of tagging each word in a phrase with one of five semantic roles: *modifier* (m), which describe the quality of a head word, such as "mobile" and "personal;" *thing* (t), which is a concept that has logical boundaries and can be composed of other things; *event* (e), which describe action performances, such as "usage," "viewing," and "clicks;" *agent* (a), which describe actors who perform actions or possess things; *property* (p), which describe the functional feature of an agent, place or thing such as "date," "name," "height;" and (x) which is an *abstract tag* indicating any general category of information, including "information," "data," and "details," among others. In an ontology, the concept that corresponds to x

(e.g., "information") is the most general, inclusive concept in the hierarchy [21]. The roles are the result of grounded analysis on lexicon L_1 conducted by Hosseini et al. [21].

Part-of-speech (POS) is commonly used to tag natural language phrases and sentences [24]. *event* (e) words, for example, often correspond to noun-forms of verbs with special English suffixes (e.g., "usage" is the noun form of "use" with the suffix "-age"), and *things* (t) and *actors* (a) are frequently nouns. However, the analysis of lexicon L_1 shows that only 22% of tagged sequences can be identified using POS and English suffixes [21]. Therefore, we rely on manual tagging of words using five semantic roles by two analysts. The effort required for this task is linear in the size of lexicon.

The information type tagging is expressed as a continuous series of letters that correspond to the semantic roles. Figure 3 shows an example information type, "mobile device identifier" that is decomposed into the atomic words: "mobile," "device," and "identifier," and presented with tag sequence mtp. The intuition behind step 2 in the overall approach is based on the observation that information types are frequently variants of a common lexeme.

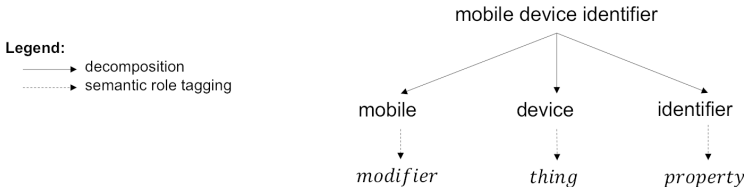

Fig. 3. Example of Lexicon Phrase, Tokenized and Tagged

4.3 Syntactic Analysis of Information Types Using Context-Free Grammar

A context-free grammar (CFG) is a quadruple $G = \langle N, V, R, S \rangle$, where N, V, and R are the sets of non-terminals, terminals, productions, respectively and $S \in N$ is the designated start symbol.

Step 3 (Figure 2) begins by processing the tagged information types from the reduced lexicon using the CFG in Table 1. The CFG represents the antecedent and subsequent tags used to infer morphological variants from a given information type. The grammar is yielded by applying grounded analysis to the tag sequences of all information types in lexicon L_1. Notably, the grammar distinguishes between four kinds of tag sub-sequences: (1) a type that is modified by a modifier, called *Modified1*; (2) a type that is modified by an agent (e.g., "user" or "company") or event (e.g., "click" or "crash"), called *Modified2*; (3) a *Final* type that describes the last sequence in a typed string, which can end in a part, an information suffix, or an empty string; (4) for any parts of a whole (*Part*), these may be optionally described by modifiers, other parts, or things; and (5) *Info*, including those things that are described by information (e.g., "device information").

Figure 4 shows the parse tree for the phrase "mobile device identifier" with type sequence mtp. Next, we discuss how these productions are extended with semantic attachments to infer ontological relationships.

<*S*>→<*Modified1*> | <*Modified2*> | <*Final*> | x
<*Modified1*>→ m<*Modified1*> | m <*Modified2*> | m <*Final*> | mx
<*Modified2*>→ a <*Final*> | e <*Final*> | a <*Info*>
<*Final*>→ t <*Part*> | t <*Info*> | e <*Info*> | p
<*Part*>→<*Modified1*> | <*Modified2*> | <*Final*>
<*Info*>→ $x\|\epsilon$

Table 1: Context-Free Grammar for Syntax Analysis

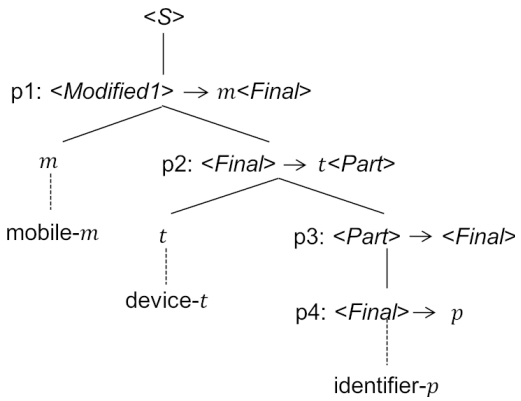

Fig. 4. Parse Tree for "mobile device identifier" with Tag Sequence "mtp"

4.4 Inferring Morphological Variants and Semantic Relations

Based on the compositionality principle, the meaning of a sentence can be constructed from the meaning of its constituents [15,23]. We adapt this principle to infer semantics between an information type and its constituent morphological variants by extending the CFG production rules with semantic attachments.

Each production $r \in R, r : \alpha \rightarrow \beta_1 ... \beta_n$ is associated with a semantic rule $\alpha.sem : \{f(\beta_1.sem, ..., \beta_n.sem)\}$. The semantic attachment $\alpha.sem$ states: the representation assigned to production r contains a semantic function f that maps semantic attachments $\beta_i.sem$ to $\alpha.sem$, where each $\beta_i, 1 \leq i \leq n$ is a constituent (terminal or non-terminal symbol) in production r. The semantic attachments for each production rule is shown in curly braces $\{\dots\}$ to the right of the production's syntactic constituents. Due to space limitations, we only present the semantic attachments of four production rules used in Figure 4 in Table 2. The full table is published online [5]. We first introduce λ-calculus functions used in Table 2, before presenting an example where semantic attachments are applied to the tagged information type "mobile device identifier-mtp."

In λ-calculus, functions are represented by symbolic notations called λ-expressions. *Variables* and *constants* are atomic constituents of λ-expressions. Complex λ-expressions can be built from variables based on their application and abstraction [19].

Unary function $WordOf(y)$ maps a non-terminal to its tagged phrase sequence. For example, $WordOf$(*Final*) returns "device identifier-tp" in Figure 4. In this example, *Final* refers to the left-side non-terminal of *Modifier1*.

$Concat(y, z)$ is a binary function used to concatenate two tagged phrase sequences, for example $Concat$(mobile-m, information-x) produces "mobile information-mx."

[5] http://galadriel.cs.utsa.edu/~rslavin/ontology-grammar/

$SubVariant(y)$ is a higher-order function accepting other functions like $Concat$ as an argument. It returns a list of variants that can be constructed using the input argument, e.g., $SubVariant$(mobile device identifier-mtp) returns the following list of variants: [mobile device identifier-mtp, device identifier-mtp, identifier-p].

$IsInfo$(y) is a unary function on a tagged phrase sequence, returning an empty list if the input sequence matches "information-x" and $Eqv(y$, information-$x)$, otherwise. For example, $IsInfo$(data-x) returns Eqv(data-x, information-x), since "data-x" and "information-x" do not match.

$KindOf(y, z)$, $PartOf(y, z)$, and $Eqv(y, z)$ are higher-order functions that map two tagged phrases to a single-element list containing a candidate hypernymy, meronymy, and synonymy axioms, respectively.

$Map(y, z)$ is a binary higher-order function that distributes the application of a function over a list of tagged phrases. More precisely, it can be shown as:

$Map(f, [E_1, ..., E_n]) = [(f)E_1, ..., (f)E_n]$

	Production	Semantic Attachments	Line
p1	<*Modified1*>→m<*Final*>	{$\lambda y.\lambda m.$*Final*.sem(Concat(y, m));	1
		λm.KindOf(WordOf(*Modified1*), Concat(m, information-x));	2
		KindOf(WordOf(*Modified1*), WordOf(*Final*))}	3
p2	<*Final*>→t <*Part*>	{$\lambda y.\lambda t.$ *Part*.Sem(Concat(y, t));	1
		KindOf(WordOf(*Final*), WordOf(*Part*));	2
		Map(λz.PartOf(Concat(z, WordOf(*Part*)),z))$\lambda y.\lambda$ t. SubVariant(Concat(y, t))}	3
p3	<*Part*>→<*Final*>	{λy.*Final*.sem(y)}	1
p4	<*Final*>→p	{(Map($\lambda p.\lambda z$.PartOf(p, z)))λy.SubVariant(y);	1
		$\lambda y.\lambda p$.PartOf(Concat(y,p),y)}	2

Table 2: Rules and Semantic Attachments for "mobile device identifier-mtp"

We now describe step 3 (Figure 2) using the tagged information type "mobile device identifier-mtp". The tagged information type is first parsed using the grammar in Table 1. Its semantics are computed by visiting the nodes of the parse tree in Figure 4 and applying the corresponding semantic attachments from Table 2 during a single-pass, top-down parse. Following this order, the semantics of production rule p1 is mapped to the following λ-expressions, where l in p1.l refers to line l in Table 2:

p1.1 represents an abstraction with two lambda variables, where y refers to the inherited tagged phrase from the right and top of the parse tree and m refers to the tagged phrase "mobile-m" read through the lexical analyzer. In this case, variable y refers to an empty string, since no tagged phrase precedes "mobile-m." Therefore, the first λ-expression can be reduced to *Final*.sem("mobile-m"). In this λ-expression, "mobile-m" is inherited by non-terminal *Final* in the parse tree. Based on the principle of compositionality, the semantics of a phrase depends on the order and grouping of the words in a phrase [23]. An unambiguous grammar like the CFG cannot infer all possible variants, such

as "mobile device" and "device identifier," by syntax analysis alone, because the input phrase "mobile device identifier" would require both left- and right-associativity to be decomposed into these two variants. We overcome this limitation by introducing an unambiguous right-associative grammar and utilize λ-calculus to ensure that each non-terminal node inherits the sequence of words from the node's parents and siblings.

p1.2 represents an abstraction which reduces to a list containing a semantic relation: [KindOf("mobile device identifier-mtp", "mobile information-mx")] through reading variable m from the lexical analyzer. One might raise a point that "mobile information" is not a valid phrase. We acknowledge this fact, however, applying this rule to phrases such as "unique device identifier," "anonymous device information," and "anonymous demographic information" will results in creation of "unique information," "anonymous information," and "demographic information," which are meaningful phrases. We emphasize that the variants and relations generated through our method are only *candidates* and might not be semantically sound.

p1.3 represents a λ-expression which is the application of $KindOf$ on two operands, which reduces to a single element list [KindOf("mobile device identifier-mtp", "device identifier-tp")]. In the next step, we analyze the semantics of production rule p2 that are presented using three λ-expressions:

p2.1 represents a λ-expression to concatenate tagged phrases associated with the inherited variable y and variable t and passes the concatenation result ("mobile device-mt") to direct descendants of this node.

p2.2 represents the application of $KindOf$ function on "device identifier-tp" and "identifier-p", resulting a hypernymy relation in a single element list.

p2.3 is an application that maps a λ-expression to a *list* of variants. This list is constructed using a λ-abstraction that can be reduced to SubVariant("mobile device-mt"), producing [mobile device-tp, device-t]. Finally, Map applies $PartOf$ function on all the elements of this list resulting in [PartOf("mobile device identifier-mtp", "mobile device-mt"), PartOf("device identifier-tp", "device-t")].

Without inheriting "mobile-m" from the ancestors, we would not be able to infer the meronymy relationships between "mobile device identifier-mtp" and "mobile device-mt." Moreover, variant "mobile device-mt" is generated using syntax analysis of the tagged phrase sequence and semantics attached to the syntax. In contrast, other tagged phrases like "device identifier-tp" are solely generated through syntax analysis of "mobile device identifier-mtp." By augmenting syntax analysis with semantic attachments, we capture the ambiguity of natural language as follows. If we show the grouping using parenthesis, we can present the phrase associated with "mobile device identifier-mtp" as (mobile (device identifier)) which means mobile is modifying device identifier, e.g., an IP address as a kind of device identifier that changes based on location which makes it mobile. Another possible grouping is ((mobile device) identifier) which is interpreted as an identifier associated with a mobile device, e.g., a MAC address associated with a mobile phone, tablet or laptop. Therefore, grouping of words in "mobile device identifier-mtp" helps us consider all the possible semantics associated with an ambiguous phrase.

p3.1 is used to pass the inherited tagged phrase "mobile device-mt" to *Final* as the right-hand side, non-terminal. The semantics of production rule p4 as the last node visited in the parse tree is mapped to the following attachments:
p4.1 is the application of Map to a variant list constructed from a λ-abstraction. This abstraction is reduced to SubVariant("mobile device-mt"), returning the following variant list: ["mobile device-mt", "device-t"]. Finally, Map applies $PartOf$ function on all the elements of this list resulting in [PartOf("identifier-p", "mobile device-mt"), PartOf("identifier-p", "device-t")].
p4.2 represents an abstraction that reduces to [PartOf("mobile device identifier-mtp", "mobile device-mt")].

All the above production rules and semantic attachments yield a collection of candidate relations contained in multiple lists. As the final procedure in step 3, we merge the lists and add the relations to the output ontology.

5 Evaluation and Results

We answer the following research questions as part of our evaluation:
RQ1: How much, and to what extent, does the grammar generate the relationships between information type pairs in Lexicon L_1?
RQ2: Which semantic relations are missed by the method in comparison with the ground truth ontology?
RQ3: What level of effort is required to maintain the method for each new lexicon addition, considering the type of apps and data practices the lexicon is constructed from?
RQ4: How reliable is the method with respect to a new lexicon addition?

Research questions RQ1 and RQ2 evaluate the ontology construction method using lexicon L_1, discussed in § 5.1. Research questions RQ3 and RQ4 evaluate the generalization and coverage of our method using lexicon L_2, discussed in § 5.2.

5.1 Evaluation using Lexicon L_1

We evaluate the ontology construction method using lexicon L_1 to answer RQ1 and RQ2. L_1 contains 351 information types which are used to develop the context-free grammar (CFG) in § 4.3. We acquired the reduced and tagged information types in L_1 through this link [6]. Given 335 reduced tagged information types, the CFG and semantic attachments yield 4,593 relations that share at least one common word. We published these relations in two formats[5].

We require a ground truth (GT) ontology containing the relations between information types in lexicon L_1 to evaluate the accuracy of the inferred relations to answer RQ1. We acquired the results of a study published by Hosseini et al. [21] [7] and followed their approach to construct the GT. This study contains 2,253 information type pairs which is the result of pairing all the information types that share at least one word in the reduced version of lexicon L_1 (based on step 1). Further, the study contains the relations

[6] http://gaius.isri.cmu.edu/dataset/plat17/study-platform-lexicon-typedPhrases-reduced.csv
[7] http://gaius.isri.cmu.edu/dataset/plat17/preferences.csv

assigned to each pair by 30 human subjects (called participant preferences). The participants were recruited from Amazon Mechanical Turk, had completed over 5,000 HITs, had an approval rating of at least 97%, and were located within the US [21].

Due to the diversity of participant experiences, which allows participants to perceive different phrase senses, participants can assign different semantic relations to the same pair, e.g., "mac" can refer to both a MAC address for Ethernet-based routing, and a kind of computer sold Apple. In another example, "email" can refer to three different senses: a service or program for sending messages; a message to be sent via the SMTP protocol; or to a person's email address, which is the recipient address of an email message. Therefore, participants may conclude "email address" is a part of "email", or is equivalent to "email" which are both valid interpretations. To avoid excluding valid interpretations, we follow Hosseini et al.'s approach to build a multi-viewpoint GT that accepts multiple, competing interpretations [21]. Valid interpretations for a pair are the ones that the observed number of responses per category exceeds the expected number of responses in a Chi-square test, where $p < 0.05$. This threshold means that there is at least a 95% chance that the elicited response counts are different than the expected counts [21]. The expected response counts for a relation are based on how frequently participants chose that relation across all participant comparisons. Finally, we constructed a multi-viewpoint GT as follows: for each surveyed pair, we add an axiom to the GT for a relation category, if the number of participant responses is greater than or equal to the expected Chi-square frequency; except, if the number of unrelated responses exceeds the expected Chi-square frequency, then we do not add any axioms.

We compared the inferred relations with the relations in the GT. An inferred relation is a true positive (TP), if it is logically entailed by GT, otherwise, that relation is a false positive (FP). Overall, 980 inferred relations are logically entailed in the GT. We use logical entailment to identify TPs, because subsumption is transitive and whether a concept is a hypernym of another concept may rely on the transitive closure of that concept's class relationships in the GT. We only found two inferred relations as FPs. An unrelated information type pair in the GT is considered as true negative (TN), if we cannot match any inferred relation with it. We found 805 pairs as TNs. For all information type pairs with valid interpretations (i.e., hypernymy, meronymy, and synonymy) in GT that do not match an inferred semantic relation, we count these as false negatives (FN). We found 466 of the related pairs in the GT that cannot be logically entailed in the ontology fragments inferred through our method.

We computed $Precision(Prec.) = TP/(TP + FP)$ and $Recall(Rec.) = TP/(TP + FN)$ for the ontology construction method using CFG and semantic attachments, presented in Table 3. We also compare the results of our method to the previously proposed ontology construction method using 26 regular expression patterns by Hosseini et al. [21]. Our model outperforms the 26 regular expression patterns, by decreasing the number of FNs and improving the recall.

Method	Prec.	Rec.
26 Regular Expression Patterns	0.99	0.56
CFG and Semantic Attachments	0.99	0.67

Table 3: Performance Measures for Lexicon L_1

RQ2 concerns the type of relations that cannot be inferred using our syntax-driven method. To answer this question, we open coded the 466 FNs and identified four codes that explain the reasons that our method could not infer the relations:

(1) *Tacit Knowledge:* The relation requires tacit knowledge to be inferred and may not be inferred using syntax analysis of phrases, alone. For example, the hypernymy relation between "crash events" and "device event information" requires knowing that a crash is a software or hardware failure on a device, which is tacit knowledge that our method lacks. We identified 404/466 of the FNs that fall into this category.

(2) *Parse Ambiguity:* Our method analyzes phrases by grouping words from the right and left using the CFG and inherited variants in semantic attachments, respectively. However, we have observed 17/466 of FNs that disregard this grouping and therefore, cannot be inferred by our method. For example, an equivalence relation between "device unique identifier" and "unique device identifier" would be inferred as two kinds of "device identifier," but not as equivalent concepts.

(3) *Modifier Suppression:* Participants may ignore modifier roles in a phrase and thus prefer an equivalent relation between a pair of phrases. For example, "actual location" and "approximate location" are identified equivalent in the GT ontology. This phenomenon was also reported by Hosseini et al. [21]. We identified 34/466 phrase pairs and their relations that fall into this category.

(4) *Unjustifiable:* We identified 11/466 phrase pairs in the GT that we cannot justify despite the participant preference for these relations. For example, individuals identified " general demographic information" as a kind of "general geographic information." In another example, " mobile device type" is identified as a kind of "mobile device unique identifier" by the individuals.

5.2 Evaluation using Lexicon L_2

RQ3 and RQ4 ask about the level of effort to maintain the method, and the method's reliability. We pre-processed 1,853 information types in lexicon L_2 using the strategies mentioned in § 4.1, yielding 1,693 information types. In the four steps presented in Figure 2, only step 2 involves manual effort for semantic tagging. During this step, two analysts individually assigned tags to information types in L_2. We calculated the inter-rater agreement for the assigned tags using Fleiss' Kappa co-efficient, which is a chance-corrected measure of agreement between two or more raters on a nominal scale [14]. The comparison resulted in 518 disagreements with Kappa = 0.704. After reconciling the disagreements, we increased Kappa to 0.917 and randomly selected tag assignments from one of the analysts.

To address RQ4 on method reliability, we require a ground truth for relations in L_2. For this reason, we selected information type pairs that share at least one word, yielding 1,466,328 pairs. Due to this large number, we sampled the pairs by creating strata that represent comparisons between tag sequences as follows:

Phase A: Each information type pair is mapped to their respective tag sequence pair, e.g., pair (mobile device, device name) is mapped to (mt, tp), yielding 974 unique tag sequence pairs, which we call the strata.

Phase B: Proportional stratified sampling is used to draw *at least* 2,000 samples from all strata with layer size range 1-490. The wide range in layer sizes implies unbalanced

strata; e.g., strata that contain 1-3 pairs when divided by the total number of information type pairs yields zero. Therefore, we select all the pairs from strata with size one to ensure strata coverage. For strata of size two and three, one random information type pair is selected. For the remaining strata with sizes greater than three, sample sizes are proportional to the strata size, yielding one or more pairs per stratum. For each stratum, the first sample is drawn randomly. To draw the remaining samples, we compute a similarity distance between the already selected pairs and remaining pairs in each stratum: First, we create a *bag-of-lemmas* by obtaining word lemmas in the already selected pairs. Next, in each stratum, the pairs with the least common lemmas with the bag-of-lemmas are selected. We update the bag-of-lemmas after each selection by adding the lemmas of the selected pairs. This strategy ensures the selection of pairs with lower similarity measure, resulting in a broader variety of words in the sampled set.

Further, we ensure that each tag sequence is represented by at least one sampled item, and that sequences with a larger number of examples are proportionally represented by a larger portion of the sample. Using the initial sample size of 2,000, we captured 2,283 samples from 1,466,328 phrase pairs. Our samples contain 1,138 unique information types from Lexicon L_2. Using the pairs, we published a survey that asks subjects to choose a relation for pair (A, B) from one of the following six options [21]:
s: A is a kind of B, e.g., "mobile device" is a kind of "device."
S: A is a general form of B, e.g., "device" is a general form of "mobile device."
P: A is a part of B, e.g., "device identifier" is a part of "device."
W: A is a whole of B, e.g., "device' is a whole of "device identifier."
E: A is equivalent to B, e.g.,"IP" is equivalent to "Internet protocol."
U: A is unrelated to B, e.g., "device identifier" is unrelated to "location."

We recruited 30 qualified Amazon Mechanical Turk participants following the criteria mentioned in § 5.1. We constructed a multi-viewpoint ground truth (GT) containing 2,283 semantic relations[5]. Application of the CFG and semantic attachments on sampled information types from L_2 results in 21,745 inferred relations[5]. To compute Precision and Recall, we compare the inferred relations with the multi-view GT. Overall, the method correctly identifies 1,686/2,283 of relations in the GT. We also compare the inferred relations using 26 regular expression patterns [21]with the GT. The performance measures in Table 4, suggest that our proposed CFG and semantic attachments reduce the number of false negatives (FNs). FNs are the semantic relations between information type pairs in the GT that do not match inferred semantic relations. By reducing the number of FNs, our proposed method improves the recall compared to the 26 patterns.

Method	Precision	Recall
26 Regular Expression Patterns	0.99	0.62
CFG and Semantic Attachments	0.99	0.90

Table 4: Performance Measures for Lexicon L_2

6 Threats to Validity

Internal Validity- Evaluating semantic relations depends on reliable tagging of information types by analysts. Changes in tags affect the performance of the method when

compared to the ground truth (GT). In § 5.1, we identified four categories reflecting the relations that cannot be inferred when compared with ground truth (GT) for lexicon L_1. During second-cycle coding of Tacit Knowledge category, we observed a potential explanation for why individuals prefer a relation that differs from our results. The terms in "application software" were tagged tt, which is used to entail that "software" is part of an "application." However, we believe that participants recognize that "application software" is a single entity or thing. We also believe this explanation applies to 20 phrases and 69 semantic relations in the GT. We revised the tag sequences for these phrases and inferred relations based on this revision. Applying our method on the set of revised tagged types results in an additional 74 FNs compared to the original tagged information types. For example, the method cannot infer the relations between the following pairs: ("application software," "software information"), ("page view order," "web page"). Therefore, semantic ambiguity in tokenization and tagging can result in changes to the inferred relations, which is a shortcoming of the method.

For lexicon L_2, two analysts individually assigned tags to information types with an initial Kappa = 0.70.The analysts reconciled their differences to reach a Kappa = 0.92.

External Validity- The CFG is constructed on lexicon L_1 containing 351 platform-related information types defined as "any information that the app or another party accesses through the mobile platform that is not unique to the app." The information types were extracted from collection data practices of 50 mobile app privacy policies [21,32]. To study generalizability beyond lexicon L_1, we utilize lexicon L_2 for evaluation. Lexicon L_2 contains 1,853 information types related to collection, usage, retention, and transfer data practices, extracted from 30 mobile and web app privacy policies [12]. Further study is needed to determine how well the method extends beyond these datasets.

7 Conclusion and Future Work

Privacy policies are expressed in natural language and thus subject to ambiguity and abstraction. To address this problem, we propose a method to infer semantic relations between information types in privacy policies and their morphological variants based on a context-free grammar and semantic attachments. This method is constructed based on grounded analysis of information types in 50 privacy policies and tested on information types from 30 policies. Our method shows an improvement in reducing the number of false negatives, the time, and effort required to infer semantic relations, compared to previously proposed methods by formally representing the information types. Evidence from Bhatia et al. shows that between 23-71% of information types in any new policy will be previously unseen [4], which further motivates the need for a high-precision, semi-automated method to infer ontological relationships.

In future work, we plan to augment our method with a neural network classification model to infer semantic relations that are independent of syntax and purely rely on tacit knowledge, such as hypernymy relation between "phone" and "mobile device."

Acknowledgment. This research was supported by NSF #1736209 and #1748109.

References

1. Anton, A.I., Earp, J.B.: A requirements taxonomy for reducing web site privacy vulnerabilities. Requirements Engineering **9**(3), 169–185 (2004)
2. Bach, E.: An extension of classical transformational grammar (1976)
3. Bhatia, J., Breaux, T.D.: Towards an information type lexicon for privacy policies. In: RELAW. pp. 19–24. IEEE (2015)
4. Bhatia, J., Breaux, T.D., Schaub, F.: Mining privacy goals from privacy policies using hybridized task recomposition. TOSEM **25**(3), 22 (2016)
5. Boyd, S., Zowghi, D., Gervasi, V.: Optimal-constraint lexicons for requirements specifications. In: REFSQ. pp. 203–217. Springer (2007)
6. Breaux, T.D., Antón, A.I., Spafford, E.H.: A distributed requirements management framework for legal compliance and accountability. computers & security **28**(1-2), 8–17 (2009)
7. Breaux, T.D., Baumer, D.L.: Legally "reasonable" security requirements: A 10-year ftc retrospective. Computers & Security **30**(4), 178–193 (2011)
8. Breaux, T.D., Hibshi, H., Rao, A.: Eddy, a formal language for specifying and analyzing data flow specifications for conflicting privacy requirements. RE **19**(3), 281–307 (2014)
9. Breitman, K.K., do Prado Leite, J.C.S.: Ontology as a requirements engineering product. In: Proceedings. 11th IEEE International Requirements Engineering Conference, 2003. pp. 309–319. IEEE (2003)
10. Corbin, J., Strauss, A., et al.: Basics of qualitative research: Techniques and procedures for developing grounded theory (2008)
11. De Saussure, F., Harris, R.: Course in general linguistics.(open court classics). Chicago and La Salle, Open Court (1998)
12. Evans, M.C., Bhatia, J., Wadkar, S., Breaux, T.D.: An evaluation of constituency-based hyponymy extraction from privacy policies. In: RE. pp. 312–321. IEEE (2017)
13. Fensel, D., McGuiness, D., Schulten, E., Ng, W.K., Lim, G.P., Yan, G.: Ontologies and electronic commerce. IEEE Intelligent Systems **16**(1), 8–14 (2001)
14. Fleiss, J.L.: Measuring nominal scale agreement among many raters. Psychological bulletin **76**(5), 378 (1971)
15. Frege, G.: Über begriff und gegenstand (1892)
16. FTC: Ftc's $5 billion facebook settlement: Record-breaking and history-making (2019)
17. Gervasi, V., Zowghi, D.: On the role of ambiguity in re. In: REFSQ. pp. 248–254. Springer (2010)
18. Harris, K.D.: Privacy on the go: recommendations for the mobile ecosystem (2013)
19. Henk, B.: The lambda calculus: its syntax and semantics. Studies in logic and the foundations of Mathematics (1984)
20. Hookway, C.: Peirce-Arg Philosophers. Routledge (2010)
21. Hosseini, M.B., Breaux, T.D., Niu, J.: Inferring ontology fragments from semantic role typing of lexical variants. In: REFSQ. pp. 39–56. Springer (2018)
22. Hosseini, M.B., Wadkar, S., Breaux, T.D., Niu, J.: Lexical similarity of information type hypernyms, meronyms and synonyms in privacy policies. In: =AAAI Fall Symposium (2016)
23. Janssen, T.M., Partee, B.H.: Compositionality. In: Handbook of logic and language, pp. 417–473. Elsevier (1997)
24. Jurafsky, D., Martin, J.H.: Speech and language processing, vol. 3. Pearson London (2014)
25. Massey, A.K., Rutledge, R.L., Antón, A.I., Swire, P.P.: Identifying and classifying ambiguity for regulatory requirements. In: RE. pp. 83–92. IEEE (2014)
26. Miller, G.A.: Wordnet: a lexical database for english. Communications of the ACM **38**(11), 39–41 (1995)

27. Oltramari, A., Piraviperumal, D., Schaub, F., Wilson, S., Cherivirala, S., Norton, T.B., Russell, N.C., Story, P., Reidenberg, J., Sadeh, N.: Privonto: A semantic framework for the analysis of privacy policies. Semantic Web **9**(2), 185–203 (2018)
28. Petronella, G.: Analyzing privacy of android applications (2014)
29. Reidenberg, J.R., Bhatia, J., Breaux, T.D., Norton, T.B.: Ambiguity in privacy policies and the impact of regulation. The Journal of Legal Studies **45**(S2), S163–S190 (2016)
30. Rolland, C., Proix, C.: A natural language approach for requirements engineering. In: International Conference on Advanced Information Systems Engineering. pp. 257–277. Springer (1992)
31. Saldaña, J.: The coding manual for qualitative researchers. Sage (2015)
32. Slavin, R., Wang, X., Hosseini, M.B., Hester, J., Krishnan, R., Bhatia, J., Breaux, T.D., Niu, J.: Toward a framework for detecting privacy policy violations in android application code. In: ICSE (2016)
33. Wang, X., Qin, X., Hosseini, M.B., Slavin, R., Breaux, T.D., Niu, J.: Guileak: Identifying privacy practices on gui-based data (2018)
34. Zimmeck, S., Wang, Z., Zou, L., Iyengar, R., Liu, B., Schaub, F., Wilson, S., Sadeh, N., Bellovin, S.M., Reidenberg, J.: Automated analysis of privacy requirements for mobile apps. In: NDSS (2017)