Information and Software Technology 138 (2021) 106608

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Analyzing privacy policies through syntax-driven semantic analysis of information types

Mitra Bokaei Hosseini [a,*], Travis D. Breaux [b], Rocky Slavin [c], Jianwei Niu [c], Xiaoyin Wang [c]

[a] St. Mary's University, 1 Camino Santa Maria, San Antonio, TX, United States of America
[b] Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, United States of America
[c] University of Texas at San Antonio, 1 UTSA Circle, San Antonio, TX, United States of America

ARTICLE INFO

Keywords:
Privacy policy
Ambiguity
Generality
Ontology

ABSTRACT

Context: Several government 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.

Objective: Abstract and ambiguous terminology in requirements statements concerning information types (e.g., "we collect your device information"), can reduce shared understanding among app developers, policy writers, and users.

Method: To address this challenge, we propose a syntax-driven method that first parses a given information type phrase (e.g. mobile device identifier) into its constituents using a context-free grammar and second infers semantic relationships between constituents using semantic rules. The inferred semantic relationships between a given phrase and its constituents generate a hierarchy that models the generality and ambiguity of phrases. Through this method, we infer relations from a lexicon consisting of a set of information type phrases to populate a partial ontology. The resulting ontology is a knowledge graph that can be used to guide requirements authors in the selection of the most appropriate information type terms.

Results: We evaluate the method's performance using two criteria: (1) expert assessment of relations between information types; and (2) non-expert preferences for relations between information types. The results suggest performance improvement when compared to a previously proposed method. We also evaluate the reliability of the method considering the information types extracted from different data practices (e.g., collection, usage, sharing, etc.) in privacy policies for mobile or web-based apps in various app domains.

Contributions: The method achieves average of 89% precision and 87% recall considering information types from various app domains and data practices. Due to these results, we conclude that the method can be generalized reliably in inferring relations and reducing the ambiguity and abstraction in privacy policies.

1. Introduction

Government regulations increasingly require mobile and web-based application (app) companies to standardize their data practices concerning the collection, use, and sharing of various types of information. A summary of these practices are communicated to users through online privacy policies [1,2], which have become a well-established source of requirements for requirements engineers [3,4], because they need to be consistent with software behaviors.

The challenge of acquiring requirements from data practice descriptions, however, is that privacy policies often contain ambiguities [5], which admit more than one interpretation [6]. Furthermore, policies are intended to generalize across a wide range of data practices, and are not limited to describe a single software system, in which case they also exhibit vagueness and generality [7]. Berry and Kamsties distinguish four broad categories of linguistic ambiguity, including lexical, syntactic, semantic, and pragmatic ambiguity [8]. They further separate vagueness and generality from ambiguity. *Vagueness* occurs when a phrase admits borderline cases, e.g., the word "tall" is vague when

* Corresponding author.
E-mail addresses: mbokaeihossein@stmarytx.edu (M.B. Hosseini), breaux@cs.cmu.edu (T.D. Breaux), Rocky.Slavin@utsa.edu (R. Slavin), Jianwei.Niu@utsa.edu (J. Niu), xiaoyin.wang@utsa.edu (X. Wang).

https://doi.org/10.1016/j.infsof.2021.106608
Received 15 July 2020; Received in revised form 5 March 2021; Accepted 22 April 2021
Available online 3 May 2021
0950-5849/© 2021 Elsevier B.V. All rights reserved.

considering a subject who is neither tall nor not tall [8]. In *generality*, a superordinate term refers to two or more subordinate terms. In linguistics, generality is encoded by the relationship between a *hypernym*, or the general term, and more specific terms, called *hyponyms*.

In privacy policies, information types can be expressed using both vague and general terms. Many policies describe the vague phrase "personal information", which can include both a person's "age" and their "health conditions", which users may consider more or less private, leading to boundary cases. In addition, they contain general terms, such as "address", which are intended to refer to more specific meanings, such as "postal address", "e-mail address", or "network address", in which case the reader must choose an interpretation to fit the given context. Finally, these policies also contain another kind of semantic indeterminacy that has not historically been included in ambiguity, generality or vagueness, which concerns *holonyms*, or wholes, and *meronyms*, or the parts of wholes. For example, when a policy refers to "postal address", it also refers to "city", "country", and "postal code", which are distinct parts of the postal address.

Ambiguity, generality, and vagueness have been extensively studied in requirements engineering research, particularly in regulatory and policy documents. This includes techniques to identify, classify, and model ambiguity in regulations, such as HIPAA [5,9], and techniques to identify generality [3,10,11] and vagueness [12] in privacy policies. Recently, two studies employed hand-crafted regular expressions over nominals, and constituency parse trees derived from individual policy statements to extract generalities, specifically hypernyms [13,14]. This prior work demonstrates the difficulty of scaling manual methods to construct ontologies from policies, and thus motivates the need for automated ontology discovery techniques.

In this paper, we focus on the role of hypernyms, meronyms and synonyms and their formal relationships among terminology in privacy policies (see Section 2 for examples). We propose a novel, automated syntax-driven semantic analysis method for constructing partial ontologies to formalize these relationships. Formal ontologies can be used to automate requirements analysis, specifically where the "informal meets the formal", as in where mathematical models are extracted from natural language text [15]. Recently, such ontologies have enabled precise, reusable and semi-automated analysis to trace requirements from policies to code execution [11,16] and in checking formal specifications for conflicting interpretations [3,17].

Our proposed method is based on the principle of compositionality, which states the meaning of a given phrase can be derived from the meaning of its constituents [18,19]. Using this principle and grounded analysis of 356 unique information type phrases (e.g., mobile device identifier), we developed a context-free grammar (CFG) to decompose a given information type phrase into its constituents. The production rules in the CFG are augmented with semantic rules, which we call semantic attachments [20], that are used to infer semantic relationships, including *hypernymy*, *meronymy*, and *synonymy* between the given information type constituents. This method is evaluated on two sets of 491 and 1853 information type phrases extracted from 60 privacy policies. Applying our method on these information types yields 5044 and 21,745 semantic relations, respectively.

This work extends a previous conference paper [21] with new evaluation of the syntax-driven method using two sources of ground-truth: (a) relations identified by experts with experience in privacy and data practices; and (b) the preferences expressed by a population of web and mobile-app users (i.e., non-experts) toward relationships between information types. This novel contribution adds a new evaluation method that reaches beyond expert opinion, which is the historical benchmark for constructing corpora and performing natural language evaluation, to include popular opinion, which is better suited to measure how potential users interpret data practice descriptions in privacy policies. The overall contributions of the current paper are as follows: (1) a syntax-driven method to infer semantic relations from a given information type using principle of compositionality; (2) an empirical evaluation of the method using expert-inferred relations; (3) an empirical evaluation of the method using population preferences; (4) an empirical evaluation of the method using statements about mobile and web-based apps across multiple domains.

This paper is organized as follows: in Section 2, we discuss the problem and motivation for ontology construction with an example ontology illustration; in Section 3, we discuss important terminology and lexicons; in Section 5, we introduce the syntax-driven method. In Section 6, we present the evaluation using four experiments and results, followed by limitations in Section 7, threats to validity in Section 8, and related work in Section 4. Finally, we discuss the application and implication of ontologies in practice, and future work in Sections 9 and 10.

2. Problem and motivation

Ambiguous terminology in legal requirements and privacy policies can lead to multiple, unwanted interpretations [9] and further reduce the shared understanding among requirements engineers, policy authors, and regulators [22]. A lack of shared understanding has consequences, such as the recent $5 billion settlement of Federal Trade Commission with Facebook [23]. This penalty arose from poor data practices resulting in the leaking of 87 million users' personal information to third parties. The best intent to comply with laws can be obstructed by ambiguous policies. In general, ambiguity in privacy policies creates challenges for managing privacy, tracing privacy requirements to data practices in code, and checking the compliance of privacy policies with regulations.

Researchers have proposed methods to ensure data transparency and compliance by analyzing data practices expressed in privacy policies. For example, Breaux et al. formalized data practice requirements from privacy policies using Description Logic [24] to automatically detect conflicting requirements [3] and to trace data flows across policies of interacting services [3]. Tracing privacy requirements across policies can enhance developers' understanding of third-party data use, and assist in compliance 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 [25,26].

These methods are based on manually-compiled "flat" lexicons, wherein information types were grouped into a small number of categories, such as "location", "contact", or "identifier" [26]. Lexicons that group terms into a small number of broad categories introduce inaccuracies because categorization is based on satisficing, in which the analyst ignores subtle differences between the meanings of two or more terms. For example, the phrase "WiFi SSID" can be construed to mean a type of location information [26] when location is one of a few categories to choose from (i.e., the categorization is a good fit, given the constraints). However, the term actually describes a compound technology that must be combined with the MAC address used by the device that is referenced by the SSID, before it can be indirectly correlated with location. Without this technology, SSIDs are not a kind of location information, whereas city, zip code, and longitude and latitude are kinds of location information.

In addition to coarse categories, ambiguity and inconsistency arise from *hypernymy*, which occurs when a more 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") [27].

Consider the following snippet from EA Games' privacy policy[1] 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; [...];

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

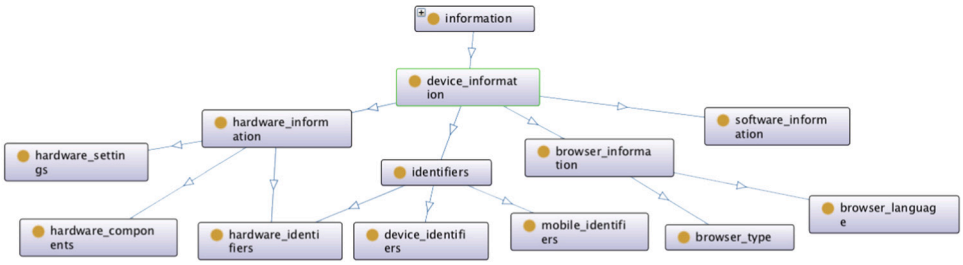

Fig. 1. Example ontology.

Information about your device, hardware and software, such as your hardware settings and components [...]". In this example, an analyst who is compiling this list within their company 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 similar hypernymy relationships between information types intuitively by applying their domain knowledge and experience. Moreover, an information type can have more than one hypernym. An analyst documenting these inferences could create a reusable ontology, shown in Fig. 1, to make the semantic relationships among terms explicit via hypernymy.

Ontology Usage Scenario: In our previous research, we developed the policy analysis tool `PVDetector` (Privacy-policy Violation Detector) to identify misalignments between policy and practice [28]. This tool maps the descriptive information types in privacy policies expressed in natural language to privacy-related API methods implemented in the corresponding code. This mapping provides the semantics needed to check code for misalignment with privacy policy, and to suggest where the code or policy may be changed to fit the functional and legal requirements of apps [28]. During misalignment detection, the tool utilizes an ontology of private information terms as a resource to formalize and identify relations between information types in natural language privacy policies. `PVDetector` is designed to be fully automatic for arbitrary Android apps given an up-to-date policy term ontology and API-phrase mapping. The input of `PVDetector` is the app's byte code (i.e., apk file) and corresponding privacy policy. The output is a list of detected misalignments which are marked as either *strong* (i.e., no information type related to an API method invocation is found in the privacy policy) or a *weak* (i.e., no information type *directly mapped* to an API method invocation is found, but some related abstract information type is found in the privacy policy). For example, assume an app has the following text in its privacy policy. "*We collect browsing history, contact information,* ***mobile identifiers****, and language information.*" If, upon analysis of the app's actual dataflows, a flow is detected from the method `getMacAddress()` to some method that sends data away through the network, the possibility of private information leakage implied. Therefore, some corresponding information type (e.g., "MAC address") should appear in the privacy policy as information that may be collected. Instead, the more broad information type "mobile identifiers", which does not directly describe "MAC address" but is *intuitively* representative of it, appears in the policy. Without considering the hypernymy relation between "MAC address" and "mobile identifiers" (i.e., MAC address is a kind of mobile identifiers), an incorrect misalignment of omission by the policy would be detected (i.e., false positive). To remedy this, an ontology containing such relationships facilitates the detection of the hypernym in the policy. Thus, the false negative due to the use of a broader term in the policy is avoided, and the policy writer can be made aware of the more specific term "MAC address" for improved clarity and precision.

In order to support automated analysis of requirements, tools and techniques are needed to build and validate formal ontologies that encode what analysts know using their domain knowledge and experience. In this paper, we propose a methodology to create ontologies similar to the one shown in Fig. 1. In the next section, we discuss important background and terminology, before introducing the syntax-driven method to infer semantic relationships from information type phrases.

3. Background

In this section, we introduce the terminology and three different lexicons used throughout this paper.

3.1. Terminology

Hypernymy: a relationship between two noun phrases where the meaning of one phrase, called the hypernym, is more generic than the other phrase, called the hyponym, e.g., "device information" is a hypernym of "device identifier".
Meronymy: a relationship between one phrase, called the whole, and a second phrase, which is a part of the whole, e.g., "device identifier" is a part of "device".
Synonymy: a relationship between two noun phrases with a similar meaning or an abbreviation, e.g., "IP" is a synonym of "Internet protocol".
Population Preferences: In psychology, preferences reflect an individual's attitude toward one or more objects, including a comparison among objects [29]. In this paper, we evaluate the semantic relationships inferred by the syntax-driven method using preferences expressed by a population of web and mobile-app users (i.e., non-experts).
Expert-Inferred Relationships: Semantic relationships among information types identified by privacy experts, including the authors of this paper.
Lexicon: a collection or list of information type phrases.
Ontology: an arrangement of concept names in a graph in which terms are connected via edges corresponding to hypernymy, meronymy and synonymy relationships [30]. In this paper, we only consider information type names as concept names.

In this paper, we use the following formal representation for an ontology. An ontology is a knowledge base KB expressed using $\mathcal{FL}_0$, a sub-language of the Attribute Language ($\mathcal{AL}$) in Description Logic (DL). A DL knowledge base KB is comprised of two components, the $TBox$ and the $ABox$ [31]. The $TBox$ consists of terminology, i.e., the vocabulary (concepts and roles) of an application domain. The $ABox$ contains assertions about named individuals using this vocabulary. Each ground-truth ontology knowledge base KB (i.e., an ontology either constructed from populations preferences, or expert-inferred relationships and used as a gold standard for our method evaluation) only contains terminology, which we call the $TBox\mathcal{T}$.

The semantics of $\mathcal{FL}_0$-concepts begins with an *interpretation* $\mathcal{I}$ that consists of a non-empty set $\Delta^{\mathcal{I}}$ (the domain of the interpretation) and an interpretation function, which assigns to every atomic concept C, a

Table 1
List of Lexicons.

Lexicon ID	Information types	App type	Domain(s)
L_1	Platform	Mobile	Unrestricted
L_2	User-provided	Mobile	Finance, health, online dating
L_3	Any	Mobile & Web	Shopping, telecom., social media, employment, health, and news

set $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$. The $TBox\mathcal{T}$ also contains *terminological axioms* that relate concepts to each other in the form of subsumption and equivalence, which we use to formalize hypernymy, meronymy, and synonymy. A concept C is subsumed by a concept D, written $\mathcal{T} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$ that satisfy the $TBox\mathcal{T}$. We define meronymy relationship using a role called $PartOf$ and subsumption relationship as follows: $C \sqsubseteq \exists PartOf.D$.

The concept C is equivalent to a concept D, written $\mathcal{T} \models C \equiv D$, if $C^{\mathcal{I}} \equiv D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$ that satisfy the $TBox\mathcal{T}$. Axioms of the first kind ($C \sqsubseteq D$) are called inclusions, whereas axioms of the second kind ($C \equiv D$) are called equalities [31]. Note that the equalities $C \equiv D$ can be rewritten as two inclusion axioms $C \sqsubseteq D$ and $D \sqsubseteq C$.

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 possible 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 [32]. Peirce defines a similar relationship among sign-vehicles, objects, and interpretants [33].

Context-free Grammar: a set of production rules, expressing the way that symbols of a language can be grouped and ordered together [30].

Semantic Attachments: a set of rules that augment the production rules. These attachments are instructions that specify how to compute the meaning representation of a construction from the meaning of its constituents parts [30].

3.2. Lexicons

In this paper, we refer to three privacy policy lexicons, which contain a list of information type phrases that were manually extracted from privacy policies. We use the information types in these lexicons to infer semantic relationships through our method. In this section, we provide detailed information regarding these three lexicons presented in Table 1. Each lexicon is assigned to a unique ID and can be distinguished by (1) the information type that analysts were asked to extract from the source policies, (2) the app type governed by the policy (i.e., mobile or web apps); and (3) the domains predominantly supported by the apps. Given the diversity of information types, app types, and domains, these three lexicons represent a diverse cross-section of app behavior.

Lexicon L_1 was published by Hosseini et al. [11,13], and contains 356 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 the data collection practices of 50 mobile app privacy policies [11,13]. These 50 policies were unrestricted by domain, and thus cover a wide range of domains, including gaming, finance, communication, music, productivity, social and entertainment, sports, and shopping, to name a few. This lexicon is available online [34].

Lexicon L_2 was published by Wang et al. and contains 491 user-provided information types (e.g., "user's weight", "credit card number", and "sexual orientation") defined as "any information that users explicitly provide to the app or other party" or "any information that the app or other parties collect or access that are unclear about whether the information is provided by the user or by automatic means" [16]. These types were extracted from the top 30 mobile apps in Google Play in finance (personal budget and banks sub-domains), health (personal health, insurance-pharmacy sub-domains), and dating (serious and casual dating sub-domains) domain. This lexicon is available online [34].

Finally, **Lexicon** L_3 was published by Evans et al. and contains 1853 information types related to any data collection, use, retention, and sharing practice [14]. These types were extracted from 30 mobile and web app privacy policies across six domains (shopping, telecommunication, social networks, employment, health, and news). This lexicon is available online [34].

Domains further affect the types of information described in policies. In Table 2, we present the top 10 most frequent information types across each lexicon.

4. Related work

In this section, we review related work concerning the definition of ambiguity, tools and techniques to detect and resolve ambiguity, ontology in requirements modeling, and ontology in requirement analysis. Lastly, we summarize our observation over the related work and make our contribution explicit to the body of knowledge in requirements engineering domain.

4.1. Definition of ambiguity

According to IEEE Std 830-1998 [35], a requirement is ambiguous if it admits more than one interpretation. Harwell et al. [36] defines a requirement as unambiguous if different stakeholders with similar backgrounds give the same interpretation to it. Privacy policies contain privacy requirements that are subject to different interpretations between requirement engineers, policy authors, and end-users due to their background knowledge and experience. In general, ambiguity is widely categorized as linguistic ambiguity and RE-specific ambiguity [6]. According to Kamsties and Paech, a requirement is specifically RE ambiguous, if it allows several interpretations with respect to other requirements, the application domain (e.g., applicable standards, operational environment), and the system domain (e.g., conceptual models of software systems and their behavior) [6].

Berry and Kamsties distinguish four broad categories of linguistic ambiguity, including lexical, syntactic, semantic, and pragmatic ambiguity [8]. *Lexical ambiguity* occurs when a word has several meaning (e.g., homonymy and polysemy) [8]. *Syntactic ambiguity* occurs when a given sequence of words can be given more than one grammatical structure, and each has a different meaning (e.g., in compiler construction, syntactic ambiguity occurs when a sentence has more than one parse) [8]. *Semantic ambiguity* occurs when a sentence has more than one way of reading it within its context although it contains no lexical or syntactic ambiguity [8]. Semantic is concerned with context-invariant meaning. *Pragmatic ambiguity* occurs when a sentence has several meanings in the context in which it is uttered [8]. Pragmatics is concerned with context-dependent meaning. Berry and Kamsties also discuss two phenomena closely related to ambiguity, including vagueness and generalization [8]. A requirement is vague if it is not clear how to measure whether the requirement is fulfilled or not (e.g., fast response time, user-friendly). A requirement is general if it has continuum of possible interpretations, a general meaning that covers these readings is available, but borderline cases do not exit, and it can be made precise [6] (e.g., collection of device information). For subcategories and detailed examples, we urge the reader to refer to [6,8]. In another categorization by Massey et al. [9], generality of terms in RE is treated as part of lexical ambiguity.

Table 2
10 most frequent information types in Lexicons L_1, L_2, and L_3.

Lexicon L_1		Lexicon L_2		Lexicon L_3	
Info Type	Frequency	Info Type	Frequency	Info Type	Frequency
IP address	41	Personal information	317	Information	1098
Browser type	21	Information	200	Personal information	463
IP addresses	21	Data	41	Cookies	180
Internet protocol	16	Email address	36	Personally identifiable information	165
Location information	16	Name	35	Name	100
Device identifiers	14	Health information	27	Protected health information	72
Device type	13	Protected health information	27	Email address	71
Information	12	Contact information	21	Data	59
Device information	11	Personal identifiable information	20	Contact information	41
Location	11	Personal identifiable information	19	CPNI	41

4.2. Detection and resolution of ambiguity

Detection and resolution of ambiguity in natural language requirements has been the subject of studies in the requirement engineering community for decades.

Lami et al. [37] present a tool, called QuARS, for analyzing natural language requirements. The tool allows the requirements engineers to perform an initial parsing of the requirements for automatically detecting potential linguistic defects that can determine ambiguity. The tool is composed of terms and linguistic constructions characterizing a particular defect. The categories of language defects detected by this tool does not contain generalization and differs from established ambiguity categories defined by [6,8,9]. Gleich et al. [38] propose a tool for identifying lexical, semantic, syntactic, pragmatic ambiguity, and vagueness and language errors. This tool relies on part of speech tagging and set of regular expressions to identify ambiguity. Nigam et al. [39] develop a tool to review requirements by identifying ambiguous words and provide the possible sources of wrong interpretation. The tool supports identification of lexical, syntactic, and syntax ambiguities and utilizes parts of speech tagger and a corpus of ambiguous words. Mich and Garigliano [40] calculate the ambiguity index of word as a weighted function of (1) the number of semantic meanings of and (2) the number of syntactic roles for the word. The weights depend on frequencies of occurrences for different meanings and roles in the language of the requirement statements [41]. Kiyavitskaya et al. [41] take into account the classification of ambiguity by Berry and Kamsties [8] and develop two free tools focused on (1) measuring lexical and syntactic ambiguities, and (2) identifying specific instances of pragmatic, software-engineering, and language-error ambiguities in sentences. The first tool analyzes the parse tree of a given requirement statement for structural issues and calculates the ambiguity index using the method proposed by [40]. The second tool applies parsing, part of speech tags, and WordNet to identify instances of pragmatic, software-engineering, and language-error ambiguities. Yang et al. [42,43] investigate anaphora, also called referential, ambiguity as a type of pragmatic ambiguity in requirement documents. They identify heuristics leading to nocuous and innocuous anaphoric ambiguities through grounded analysis of requirement documents. The heuristics are further employed in a machine learning classification tool to automate the anaphora ambiguity detection. Ferrari and Gnesi [44] present an approach to detect pragmatic ambiguities, where the meaning of a sentence is dependent on the context it is used. The approach relies on n knowledge graphs that are built upon n individual requirement documents in a specific domain. A knowledge graph for an individual document formalizes word stems as nodes and stem co-occurrences in sentences as edges. Given a new requirement sentence, the proposed algorithm first constructs the stem pairs and detects shortest paths that connects the paired stems in each knowledge graph. The nodes within the identified n shortest paths are then compared using a similarity measure. Comparison of the similarity measure with a pre-defined threshold signals potential pragmatic ambiguity in a given requirement statement. The knowledge graphs in this work model the contextual distance of stemmed words. However, the knowledge graphs are not designed to formalize the semantic relations between phrases. Ferrari et al. [45] define a framework to identify and categorize ambiguity in requirements elicitation interviews. This framework is built upon a set of customer-analyst interviews. Their work also addresses the consequences of ambiguity on the loss of tacit knowledge from customer to analyst during interviews. Massey et al. [5] propose a manual method for modeling legal text alongside models of software requirements. Using this method, analysts classify different elements in regulations, such as HIPAA, using ambiguity taxonomy introduced by Massey et al. [9]. Further, the analyst documents the ambiguity for ambiguous elements and models the unambiguous elements using traditional requirements modeling techniques.

Lexicons play an important role in reducing ambiguity and improving the quality of specifications [46]. Boyd et al. proposed to reduce ambiguity in controlled natural languages by optimally constraining lexicons using term *replaceability* [47]. 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. Bhatia et al. [12] introduce a theory of vagueness for privacy policy statements based on a taxonomy of vague terms derived from an empirical content analysis of 15 privacy policies. Their taxonomy is evaluated in a paired comparison experiment and results were analyzed to yield a rank order of vague terms in both isolation and composition. This theory predicts how vague modifiers to information actions and information types can be composed to increase or decrease overall vagueness.

Lexical ontologies, such as WordNet, are used in detecting ambiguity [41]. WordNet is a lexical ontology containing English words grouped into nouns, verbs, adjectives, adverbs, and function words [48,49]. Within each category, the words are organized by their semantic

relations, including hypernymy, meronymy, and synonymy [49]. WordNet is reported as the most utilized lexicon to support NLP-related RE tasks [50]. However, an analysis by Hosseini et al. [10] reveals that only 14% of information type phrases from a privacy policy lexicon are found in WordNet, mainly because the lexicon is populated with multi-word, domain-specific phrases. Therefore, finding an information type phrase can be a challenging task for requirement analysts. We aim to address this limitation and facilitate automated analysis of data requirements. Evans et al. [14] applied an extended set of 72 Hearst patterns to privacy policies to extract hypernymy pairs. Pattern sets are limited because they must be manually extended to address new policies. Hosseini et al. [13] proposed 26 regular expression patterns to parse the information types in lexicon L_1 (see Section 3) and to infer semantic relations based on their syntax to address generalization in privacy policy terminology. 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.

4.3. Requirements modeling

Ontologies are a standard form for representing the concepts within a domain, as well as the relationships between those concepts in a way that allows automated reasoning [51]. Due to such benefits, prior work in RE has employed ontology in requirements formalization and modeling.

Gordon and Breaux [52] proposed a manual framework to reconcile and formalize regulatory requirements from multiple jurisdictions into a single standard of care. The framework preserves traceability so that a business analyst can trace observed similarities and differences from requirements to specific sentences and phrases in the law. This includes formalizing specific changes in generality across related requirements. Breitman and do Prado Leite describe how ontologies can be used to analyze web application requirements [17]. Breaux et al. use an ontology to identify conflicting requirements across vendors in a multi-stakeholder data supply chain [3]. 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 [53]. 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. Humphreys et al. [54] propose a model to semi-automatically populate legal ontologies by extracting definitions, norms, and other elements of regulations semantic role labeling. This work emphasizes ontology reuse in knowledge management systems. In summary, Dermeval et al. [51] provide a systematic review of the literature in applications of ontologies in RE. Their findings provide an empirical evidence of benefits of using ontologies in RE activities both in industry and academy, to reduce ambiguity, inconsistency, and incompleteness of requirements.

4.4. Ontology application in requirement analysis

Ontologies have been applied in tracing requirements to data practices expressed in source code.

Zimmeck et al. proposed an approach to identify the misalignments between data practices expressed in privacy requirements and mobile app code [26]. 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 an ontology, this approach cannot distinguish between persistent and non-persistent information types, which afford different degrees of privacy risk to users. Harkous et al. [55] introduces a framework for privacy policy analysis, called Polisis, which aims to present privacy policy information in two usable and scalable ways; the first is a graph that visualizes the flow of user data being collected, the reasons behind the collection, and the collection choices given to users; the second is a question-and-answer system, called PriBot, that is able to answer user questions over a privacy policy. To produce these presentations, a policy is first divided into small text fragments, called segments. These segments are then classified over 10 high-level classes and 122 fine-grained classes using word embeddings, producing a set of class-value pairs for the segment. This set of class-value pairs is then used to visualize the graph or answer questions. Through this approach, each segment of the privacy policy is mapped to information types from a pre-defined set captured by Wilson et al. [56]. This approach fails to extract the exact information type from the segment and increases the abstraction level associated with each segment through the mapping, which leads to ambiguity. This work also fails to consider the introduction of new information types which do not exist in the pre-defined set.

In contrast, Slavin et al. [11] and Wang et al. [16] identify app code that is inconsistent with privacy policies using a manually constructed ontology [10]. The approach overcomes the limitation of Zimmeck et al. [26] and Harkous et al. [55] and exemplifies the efficacy of ontologies for requirements traceability. However, the manual construction of ontologies is costly and lacks scalability due to the time spent by analysts to compare information types, and errors generated by analysts during comparison [10]. Our proposed automated method is an improvement on prior work to construct ontologies that account for semantic relationships between information types in privacy policies.

5. Syntax-driven ontology construction method

Our method for constructing ontology fragments is based on grounded theory [57], which is a qualitative inquiry approach that involves applying codes to data through coding cycles to develop a theory grounded in the data [58]. Based on this theory, we describe three applications in this paper: (1) codes applied to information types in Lexicon L_1 to construct a context-free grammar (see Section 5.3); (2) memo-writing to capture results from applying the grammar and its semantic attachments to infer relations from L_1 (see experiments $E_{1.1}$ and $E_{2.1}$ in Section 6); and (3) theoretical sampling to test the proposed method on information types in lexicons L_2 and L_3 (see experiments $E_{1.2}$ and $E_{2.2}$ in Section 6).

We now describe the overview of the syntax-driven ontology construction method as presented in Fig. 2. 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.

Reproducibility. We now provide an example to illustrate the result of each step in a high-level format. Given an information type “mobile device identifiers”, in step 1, this information type is pre-processed and reduced to “mobile device identifier”. in the step 2, an analyst manually tags each word in the information type phrase. The result of this step is the information type phrase accompanied with a tag sequence “mobile device identifier-*mtp*”. Finally, in step 3, the CFG and its semantic attachments are used to extract the following candidate relationships:

- “mobile device identifier” is a hyponym (kind) of “mobile information”.

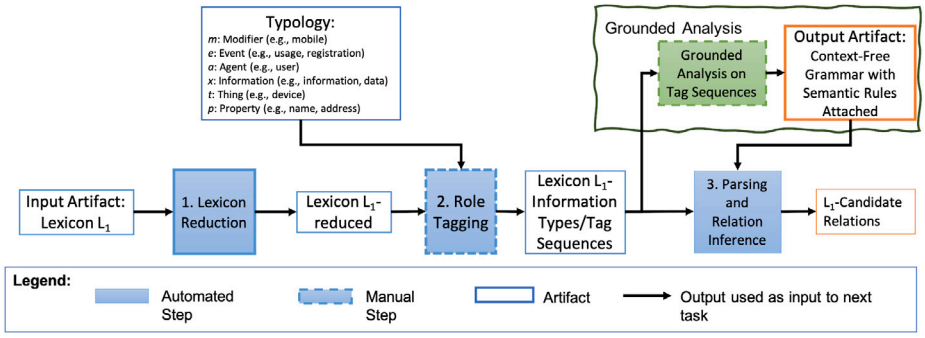

Fig. 2. Ontology construction method overview.

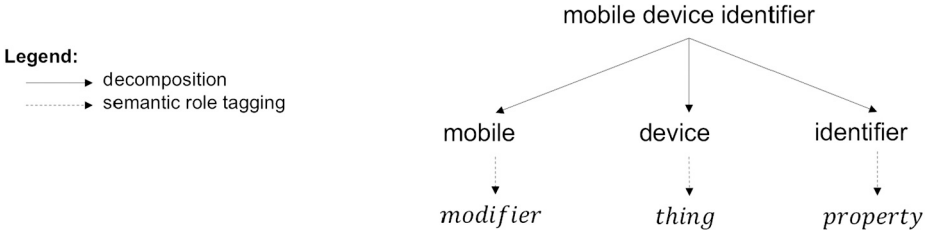

Fig. 3. Example of lexicon phrase, tokenized and tagged.

- "mobile device identifier" is a hyponym (kind) of "device identifier".
- "mobile device identifier" is a part of "mobile device".
- "device identifier" is a part of "device".
- "identifier" is a part of "mobile device".
- "identifier" is a part of "device".
- "mobile device identifier" is a part of "mobile device".

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 [20] by mapping each production rule in the CFG to its semantic counterpart, presented using λ-calculus.

5.1. 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".

5.2. 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 information type ontology, the concept that corresponds to x (e.g., "information") is the most general, inclusive concept in the hierarchy [13]. The roles are the result of grounded analysis on lexicon L_1 conducted by Hosseini et al. [13].

Table 3
Context-free grammar for syntax analysis.

$\langle S\rangle \rightarrow \langle Modified1\rangle > \mid \langle Modified2\rangle \mid \langle Final\rangle \mid x$
$\langle Modified1\rangle \rightarrow m\langle Modified1\rangle \mid m\langle\ Modified2\rangle \mid m\langle Final\rangle \mid mx$
$\langle Modified2\rangle \rightarrow a\langle Final\rangle \mid e\langle Final\rangle \mid a\langle Info\rangle$
$\langle Final\rangle \rightarrow t\langle Part\rangle \mid t\langle Info\rangle \mid e\langle Info\rangle \mid p$
$\langle Part\rangle \rightarrow \langle Modified1\rangle \mid \langle Modified2\rangle \mid \langle Final\rangle$
$\langle Info\rangle \rightarrow x \mid \epsilon$

Part-of-speech (POS) is commonly used to tag natural language phrases and sentences [30]. *Event* 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* and *actors* 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 [13]. 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, which means each new information type only needs to be tagged once.

The information type tagging is expressed as a continuous series of letters that correspond to the semantic roles. Fig. 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.

5.3. 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 (Fig. 2) begins by processing the tagged information types from the reduced lexicon using the CFG in Table 3. 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

Table 4
Rules and semantic attachments for "mobile device identifier-*mtp*".

	Production	Semantic attachments	Line
p1	$\langle Modified1\rangle \rightarrow m\langle Final\rangle$	{$\lambda y.\lambda m.Final$.sem(Concat($y$, m));	1
		λm.KindOf(WordOf($Modified1$), Concat(m, information-x));	2
		KindOf(WordOf($Modified1$), WordOf($Final$))}	3
p2	$\langle Final\rangle \rightarrow t\langle Part\rangle$	{$\lambda y.\lambda t.$ $Part$.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	$\langle Part\rangle \rightarrow \langle Final\rangle$	{$\lambda y.Final$.sem(y)}	1
p4	$\langle Final\rangle \rightarrow p$	{(Map($\lambda p.\lambda z$.PartOf(p, z)))λy.SubVariant(y);	1
		$\lambda y.\lambda p$.PartOf(Concat(y, p), y)}	2

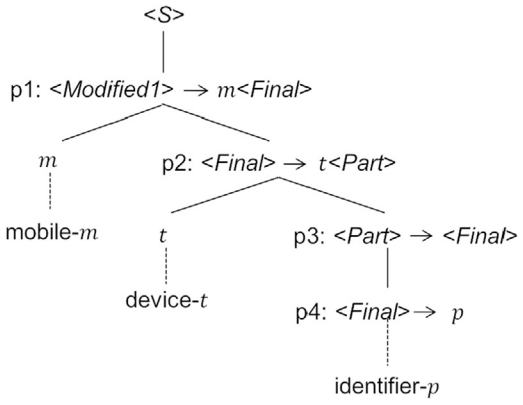

Fig. 4. Parse tree for "mobile device identifier" with tag sequence "*mtp*".

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").

Fig. 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.

5.4. 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 [18,19]. 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 $\alpha.sem : \{f(\beta_1.sem, \ldots, \beta_n.sem)\}$. 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 {…} to the right of the production's syntactic constituents. To ease the readability, we only present the semantic attachments of four production rules used in Fig. 4 in Table 4. The full table is published online [34]. We first introduce λ-calculus functions used in Table 4, 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 [59].

Unary function $WordOf(y)$ maps a non-terminal to its tagged phrase sequence. For example, $WordOf(Final)$ returns "device identifier-*tp*" in Fig. 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".

$SubVariant(y)$ 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-*tp*, 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 Equ(y, information-x), otherwise. For example, $IsInfo$(data-x) returns Equ(data-x, information-x), since "data-x" and "information-x" do not match.

$KindOf(y, z)$, $PartOf(y, z)$, and $Equ(y, z)$ 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:

$$Map(f, [E_1, \ldots, E_n]) = [(f)E_1, \ldots, (f)E_n]$$

We now describe step 3 (Fig. 2) using the tagged information type "mobile device identifier-*mtp*". The tagged information type is first parsed using the grammar in Table 3. Its semantics are computed by visiting the nodes of the parse tree in Fig. 4 and applying the corresponding semantic attachments from Table 4 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 4:

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 [19]. 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.

6. Evaluation and results

In this paper, we evaluate the syntax-driven method using two approaches:

1. Expert Evaluation: relations identified by experts with experience in privacy and data practices.
2. Non-Expert Evaluation: the preferences expressed by a population of web and mobile-app users (i.e., non-experts) toward the relationships between information types.

Fig. 5 depicts the overview of the method evaluation. Through this evaluation, we compare our method with expert and non-expert viewpoints. This comparison requires different ground truths. In the first evaluation, two ground truth sets ($KB_{1.1}$ and $KB_{1.2}$ in Fig. 5) are constructed by experts that review the information types and identify relations using seven heuristics (see Section 6.1). In the second evaluation, two new ground truth sets ($KB_{2.1}$ and $KB_{2.2}$ in Fig. 5) are constructed through a survey (see Section 6.2), capturing the preferences that reflect common population attitudes toward semantic relations among types that share at least one common word (e.g., "device identifier" and "identifier"). The experts initiate their comparisons by scanning the lexicons visually and applying pre-defined heuristics, whereas the preferences evaluation studies a complete subset of comparisons that have an above average chance of being related. In both approaches, a complete analysis of all possible pairs for lexicon of size n requires $\frac{n\times(n-1)}{2}$ comparisons, which is not feasible for large n.

In summary, we answer the following research questions with regard to the syntax-driven method.

RQ1: How much and to what extent does the method generate the relations identified by experts using a set of pre-defined heuristics?

RQ2: How much and to what extent does the method generate relations compared to population preferences?

RQ3: How reliable is the method when applied on information types extracted from *various data practices* regarding privacy policies of *mobile or web-based apps* in *various domains*?

In this section, we report the results of four experiments to address the research questions. We address research questions RQ1 by conducting two experiments $E_{1.1}$ and $E_{1.2}$ on lexicons L_1 and L_2 and their manually-constructed ontologies by experts. The results from these two experiments are discussed in Section 6.1. RQ2 aims to evaluate the method using preferences of mobile and web users population. Experiments $E_{2.1}$ and $E_{2.2}$ discussed in Section 6.2 address this research question. Research question RQ3 is addressed by examining the results of the above replications: experiments $E_{1.2}$ and $E_{2.2}$ aim to replicate the discovery of relationships extracted by experts and non-experts, respectively, while allowing the information type scope and domains to vary from the first set of experiments $E_{1.1}$ and $E_{2.1}$.

Motivation for the usage of three different lexicons: We design experiments $E_{1.1}$ and $E_{2.1}$ to evaluate the syntax-driven method and address RQ1 and RQ2 using lexicon L_1. However, the method is constructed using lexicon L_1, raising doubts regarding the *validity* of this evaluation. To provide an unbiased evaluation of the method, we design experiments $E_{1.2}$ and $E_{2.2}$ using lexicons L_2 and L_3 for ground truth construction to evaluate the method and further address research questions RQ1 and RQ2. Further, we utilize these two lexicons (i.e., L_2 and L_3) in evaluating our method's *reliability* on unseen information types from various data practices (e.g., data collection, usage, and sharing) across different app categories (e.g., shopping, finance, telecommunication, social media, etc.). Therefore, through experiments $E_{1.2}$ and $E_{2.2}$, we also evaluate the method's reliability and address research question RQ3.

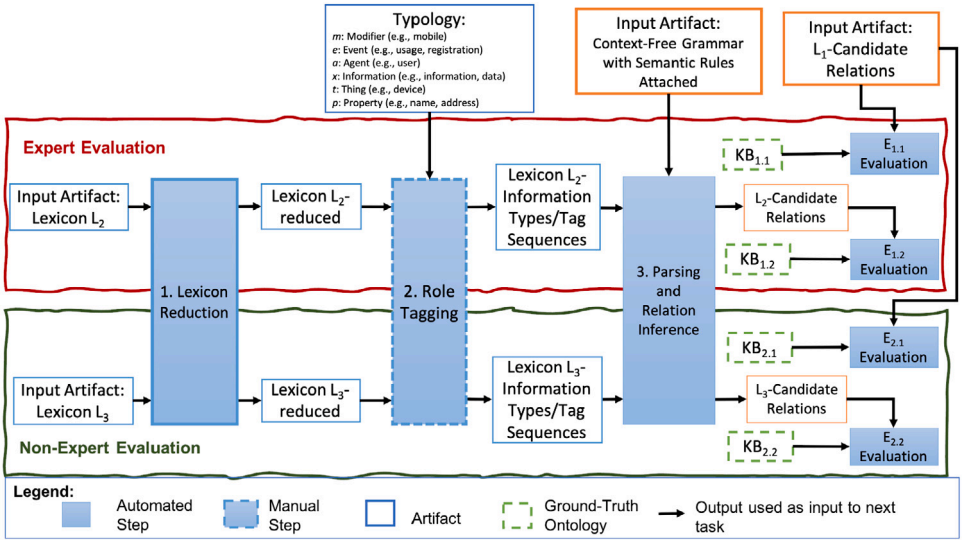

Fig. 5. Evaluation overview.

6.1. Experts inferred relations

In this section, we address RQ1 by comparing the syntax-driven method with expert-constructed ontologies created from lexicons L_1 and L_2. Experts have more extensive technical knowledge that can be used to discover relationships between information types that non-experts would otherwise miss. In addition, because expert-constructed ontologies often involve only a few experts during construction, chance-corrected statistics, such as Kappa, can be used to reduce disagreements among which relationships are valid interpretations.

Oraganization. We first describe a manual approach used to construct an ontology by privacy experts. The ontology constructed through this manual approach is used as a gold standard (i.e., ground truth) and we will compare this ontology with the semantic relations inferred by our syntax-driven method. Secondly, we explain our first experiment $E_{1.1}$ where our privacy and data practices experts utilize the manual approach to create a ground-truth ontology on the list of information types in lexicon L_1. Through this experiment, we evaluate the automatically inferred relationships from lexicon L_1 by our syntax-driven method with the ground-truth ontology and report the results of this comparison. Finally, we describe our second experiment $E_{1.2}$, where the experts utilize the manual ontology construction on lexicon L_2 to generate a ground-truth ontology. Further, we compare the inferred relationships from lexicon L_2 through our syntax-driven method with the ground-truth ontology and report the results.

6.1.1. Ground-truth ontology construction method

We now explain the manual method used by experts to construct ground-truth ontologies. Manual ontology construction begins with an initial flat ontology, wherein each information type from the lexicon is subsumed by the $\top$ concept and no other relationships exist between information types. Each expert is provided a copy of the initial ontology, which is loaded into the ontology editor Protege.[2] Next, each expert scans the alphabetical list of the information types, looking for types with which they can create a hypernymy, meronymy, or synonymy relation. Experts can search by keyword, and largely rely on their memory of information types they have seen as they discover relationships.

When an expert finds a prospect relationship, they apply one of seven heuristics that were discovered through grounded analysis of five privacy policies [10] and which are shown with corresponding axioms in Description Logic:

- Hypernym: $C \sqsubseteq D$, which means concept D is a general category of C, e.g., "password" is a kind of "authentication information".
- Meronym: C Part Of D, which means concept C is a part of concept D, e.g., "email message" is a part of "email".
- Modifiers: $C_1\_C_2 \sqsubseteq C_2$ and, $C_1\_C_2 \sqsubseteq C_1\_information$, which means concept C_1 is modifying concept C_2, e.g., "mobile phone number" is a kind of "phone number" and "mobile information".
- Plural: $C \equiv D$, which means concept C is a plural form of concept D, e.g., "addresses" is equivalent to "address".
- Synonym: $C \equiv D$, which means concept C is a synonym of concept D, e.g., "geo-location" is equivalent to "geographic location".
- Thing: $C_1 \equiv C_1\_information$, when concept C has logical boundaries and can be composed of other concepts, e.g., "name" is equivalent to "name information".
- Event: $C_1 \equiv C_1\_information$, when concept C describes an event, e.g., "usage" is equivalent to "usage information".

Finally, the expert-constructed ontologies are exported to a comparison table in which chance-agreement is measured using Fleiss' Kappa [60]. To export the relations, an algorithm extracts each expressed and inferred relationship between two information types using the Hermit[3] reasoner. This yields a table row where a relationship assigned to an information type pair appears in one column per expert.

Wherever a disagreement exists (i.e., the experts identified different relationships) the experts will review their assigned relations and discuss their disagreements with the other expert. This includes discovering missed relationships, as well as learning about new technical information to support a relationship. After the reconciliation process, the agreement is recalculated to measure the improvement due to reconciliation.

With regard to evaluation, the relations inferred by the syntax-driven method are compared to the complete entailment of the expert-constructed ontology, which includes relations inferred using the transitive closure of hypernymy. A relation found by the method is a true positive (TP), if it is logically entailed by the ontology, otherwise, that relation is a false positive (FP). If both the method and the ontology do not include a relation between two types, then that relation is counted as a true negative (TN). If the method does not find a relation that was found by an expert, then that relation is a false negative (FN). Precision is computed as TP/(TP+FP), and recall is computed as TP/(TP+FN).

[2] https://protege.stanford.edu/.

[3] http://www.hermit-reasoner.com/.

Table 5
Example table comparing information type pair relationships for Lexicon L_1.

LHS information type	RHS information type	Expert 1	Expert 2
web pages	web sites	Equivalent	Hyponym
ads clicked	usage information	Hyponym	Hyponym
computer	platform	Hypernym	Hypernym
log information	system activity	Unrelated	Hypernym
device type	mobile device type	Hypernym	Unrelated
tablet	tablet information	Unrelated	Equivalent

We now describe the two experimental results using ontologies built from the two lexicons L_1 and L_2. In these experiments, we report results for the entire expert-constructed ontology, and separate results from the ontology in which information types share at least one word. This latter result is included to support performance comparisons with the population preferences study reported in Section 6.2.

6.1.2. Experiment $E_{1.1}$: Lexicon L_1

Inferred relationships through the syntax-driven method. Lexicon L_1 contains 356 platform information types which are used to develop the context-free grammar (CFG) in Section 5.3. These information types are reduced by applying reduction strategies (see Section 5.1) to yield 347 reduced information types. Next, two analysts (i.e., the first and second authors) assign role tags to the information types as described in Section 5.2 in two consecutive rounds. The measured degree of agreement after the first round of tagging yields a Fleiss' Kappa of 0.720 and it is increased to 0.931 after reconciling disagreements in the second round [60]. The 347 reduced tagged information types are available online [34]. For the information types that share at least one word, the syntax-driven method using CFG and semantic attachments yield 4593 relations between information types, which we published here [34]. The process of inferring candidate relations from lexicon L_1 is shown in Fig. 2. In this figure, L_1 candidate relations is an output artifact that we use as an input artifact in Fig. 5.

Ground-truth ontology. To construct the ground truth for evaluation, two experts (the first and second authors) individually produce two ontologies using the methodology described above, which consist of 431 and 407 axioms, respectively. Table 5 is a snapshot of relationships assigned to information type pairs by two experts extracted from the two ontologies before any reconciliation; wherein "Hypernym" means the LHS type is a superclass of the RHS type, "Hyponym" means subclass of, "Equivalent" means equivalence to, "Whole" means the LHS is a whole that contains the LHS type as a part, "Part" means the LHS type is part of RHS type, and "Unrelated" means no relationship.

The first comparison of the two ontologies yields 321 differences measured by a 0.233 Fleiss Kappa. After two rounds of reconciliation processes, the Kappa is increased to 0.888. We utilize the reconciled version of the ontology generated by the first expert as the ground-truth ontology for experiment $E_{1.1}$, that we call $KB_{1.1}$. We list 67,289 information type pairs from $KB_{1.1}$ that are either related (i.e., hypernymy, transitive hypernymy, meronymy, and synonymy relationships) or unrelated (i.e., two information types that have no logical relationships) [34]. The statistics regarding the number of hypernymy, transitive hypernymy, meronymy, and synonymy relations, along with the unrelated pairs in $KB_{1.1}$ is presented in Table 6. Among these pairs, we select a subset of $KB_{1.1}$ with the information type pairs that share at least one common word (e.g., ⟨device, device identifier⟩). This subset of $KB_{1.1}$, called $\widehat{KB_{1.1}}$, contains 2252 information type pairs [34]. Table 6 also presents the number of pairs per category for $\widehat{KB_{1.1}}$ in comparison with $KB_{1.1}$.

Evaluation. We now evaluate the 4593 candidate relations inferred by the syntax-driven method that applies the CFG and semantic attachments to information types in lexicon L_1. These 4593 candidate relationships create fragments of an ontology when expressed in Description Logic. We first evaluate the ontology fragments against all the pairs in $KB_{1.1}$. The results for this evaluation is presented in Table 7. The syntax-driven method infers the relations in $KB_{1.1}$ with a high precision of 0.916. However, due to the high number of FNs that cannot be inferred by the syntax-driven method, the recall reaches 0.196. The syntax-driven method only infers semantic relations between information types that share at least one common word (e.g., "device" and "device identifier"). Consequently, relations between information types that rely on privacy policies' contextual semantics and experts' tacit knowledge are ignored. For example, experts identified a hypernymy relationship between "phone" and "device". However, inferring any relationship between "phone" and "device" is out of the scope of our syntax-driven method and requires additional semantic knowledge regarding these two terms. Therefore, to provide a sound comparison considering the scope of our syntax-driven method, we introduce $\widehat{KB_{1.1}}$, which contains a subset of $KB_{1.1}$ information type pairs that share at least one common word. We evaluate the inferred relations through the syntax-driven method with the relations in $\widehat{KB_{1.1}}$. Through this evaluation, 351 of the related pairs in $\widehat{KB_{1.1}}$ cannot be logically entailed in the relations inferred through the syntax-driven method. We compute precision and recall for the syntax-driven method, which is presented in Table 7. To further investigate the FNs that cannot be inferred using the syntax-driven method, we conduct an open coding study on FNs which is presented in Section 7. Finally, we compare the results of our method to the previously proposed ontology construction method using 26 regular expression patterns reported by Hosseini et al. [13]. These patterns are used to parse the information types in lexicon L_1 and to infer semantic relations based on their syntax. However, 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. Table 7 also presents the evaluation metrics for the relations inferred through the 26 regular expression patterns when compared with relationships in $\widehat{KB_{1.1}}$. Our method shows an improvement on precision and recall compared to the 26 regular expressions.

Table 6
Number of relationships in the experts-constructed ontology on L_1 (i.e., $KB_{1.1}$.

Relation	Number of pairs, $KB_{1.1}$	Number of pairs, $\widehat{KB_{1.1}}$
Direct Hypernymy	1378	404
Transitive Hypernymy	4902	205
Meronymy	327	0
Synonymy	310	109
Unrelated	60,372	1534

Table 7
Performance measures for Lexicon L_1 using expert views.

Experiment	Method	Ground-Truth Ontology	TP	FP	TN	FN	Prec.	Rec.
$E_{1.1}$	Syntax-Driven method with CFG and Semantic Attachments	$KB_{1.1}$	1347	122	60,315	5505	0.916	0.196
$E_{1.1}$	Syntax-Driven method with CFG and Semantic Attachments	$\widehat{KB_{1.1}}$	334	60	1507	351	0.847	0.487
$E_{1.1}$	26 Regular Expression Patterns	$\widehat{KB_{1.1}}$	306	59	1502	385	0.838	0.442

6.1.3. Experiment $E_{1.2}$: Lexicon L_2

RQ3 aims to evaluate the reliability of the syntax-driven method when applied on information types extracted from various data practices regarding privacy policies of mobile or web-based apps in various domains. To address RQ3, we design a second experiment with respect to expert-constructed ontologies.

Table 8
Example table comparing information type pair relationships for Lexicon L_2 (i.e., $KB_{1.2}$.

LHS Information Type	RHS Information Type	Expert 1	Expert 2
profile verification password	verification password	Subclass	Subclass
preferences	user profile information	Subclass	Unrelated
checking account number	checking account information	Subclass	Subclass
account information	routing number	Superclass	Unrelated
activity data	caloric intake	Unrelated	Superclass
address	zip code	Whole	Superclass

Table 9
Number of relationships in the experts-constructed ontology on L_2.

Relation	Number of Pairs, $KB_{1.2}$	Number of Pairs, $\widehat{KB_{1.2}}$
Direct Hypernymy	3805	138
Transitive Hypernymy	12,457	106
Meronymy	116	0
Synonymy	491	33
Unrelated	350,879	1684

Inferred relationships through the syntax-driven method. This experiment uses a previously published lexicon L_2 containing 491 user-provided information types extracted from collection data practices of 30 finance, health, and dating mobile app privacy policies [16]. As shown in Fig. 5, in step 1, we reduce the information types using the reduction strategies in Section 5.1, yielding a total number of 464 reduced information types. In step 2, two analysts (i.e., the first and second authors) apply role tagging (see Section 5.2) on the information types in two consecutive rounds. The degree of agreement is measured after each round, resulting in Fleiss' Kappa 0.656 and 0.878, respectively [60]. Given the reduced tagged information types as the input to step 3, the syntax-driven method yields 5044 relations between information types that share at least one word, which is published here [34]. This process results in L_2 candidate relations in Fig. 5.

Ground-truth ontology. In this experiment, we utilize an ontology manually built from lexicon L_2 by privacy experts as the ground truth. Two experts individually produce two ontologies using the methodology described above, which consist of 2590 and 2791 axioms, respectively. Table 8 is a snapshot of relationships assigned to information type pairs by two experts extracted the two ontologies before any reconciliation. Prior to any reconciliation, the first comparison of the ontologies yields Fleiss' Kappa of 0.590. After reconciliation the Kappa is increased to 0.836 [16,60].

We utilize the reconciled version of the ontology constructed by the first expert as the ground truth expert-constructed ontology for lexicon L_2, which we call $KB_{1.2}$. This manually constructed ontology can be retrieved here [34]. We list 367,748 information type pairs from $KB_{1.2}$ that are either related (i.e., hypernymy, transitive hypernymy, meronymy, and synonymy relationships) or unrelated (i.e., two information types that have no logical relationships) [34].

The statistics regarding the number of hypernymy, transitive hypernymy, meronymy, and synonymy relations, along with the unrelated pairs in $KB_{1.2}$ are presented in Table 9. Among these pairs, we select a subset of $KB_{1.2}$ with the information type pairs that share at least one common word (e.g., ⟨account balance, account contact information⟩). This subset of $KB_{1.2}$, called $\widehat{KB_{1.2}}$, contains 1961 information type pairs [34]. Table 9 also presents the number of pairs per each category for $\widehat{KB_{1.2}}$ in comparison with $KB_{1.2}$.

Evaluation. The results of the second experiment $E_{1.2}$ presented in Table 10 assess the method's reliability and address RQ3. We first evaluate the ontology fragments inferred by the syntax-driven method against all 367,748 pairs in $KB_{1.2}$. Similar to experiment $E_{1.1}$, the evaluation exhibit high precision and low recall due to the number of pairs in $KB_{1.2}$ that do not share any common words.

We also evaluate the syntax-driven method and the inferred ontology fragments using $\widehat{KB_{1.2}}$. We find 44 of the related pairs in the $\widehat{KB_{1.2}}$ that cannot be logically entailed in the ontology fragments inferred through the syntax-driven method. We compute precision and recall for the syntax-driven method, which is presented in Table 10. Similar to experiment $E_{1.1}$, we investigate relations identified as FNs by conducting an open coding study, which is presented in Section 7. In comparison to experiment $E_{1.1}$, lexicon L_2 yields a higher recall by 0.353, with nearly equivalent precision. Finally, we compare the results of our method to the previously proposed ontology construction method using 26 regular expressions reported by Hosseini et al. [13]. Our method outperforms the 26 regular expression patterns, by decreasing the number of FNs and improving the recall significantly.

Table 10
Performance measures for Lexicon L_2 using expert views.

Experiment	Method	Ground-Truth Ontology	TP	FP	TN	FN	Prec.	Rec.
$E_{1.2}$	Syntax-Driven method with CFG and Semantic Attachments	$KB_{1.2}$	3928	369	350,560	12,891	0.914	0.233
$E_{1.2}$	Syntax-Driven method with CFG and Semantic Attachments	$\widehat{KB_{1.2}}$	232	58	1627	44	0.800	0.840
$E_{1.2}$	26 Regular Expression Patterns	$\widehat{KB_{1.2}}$	125	38	1648	150	0.766	0.454

6.2. Population preferences

In this section, we describe the design of two experiments (i.e, $E_{2.1}$ and $E_{2.2}$) to address RQ2. These experiments compare the results from the syntax-driven method to ground truths consisting of population preferences, which are preferences from non-experts about whether two information types are related.

Organization. We first describe our general survey design published on Amazon Mechanical Turk (AMT) to capture users' preferences and interpretations toward the relationships between information types. Users' preferences (i.e., relationships between information types) are then analyzed and formalized using DL axioms to construct ground-truth ontologies. Secondly, we describe experiment $E_{2.1}$, where we compare a ground-truth ontology built upon information types in lexicon L_1 with the relationships automatically inferred by our syntax-driven method. We then report the result of this comparison. Finally, we describe experiment $E_{2.2}$, where we compare a ground-truth ontology built upon information types in lexicon L_3 with the relationships automatically inferred by our syntax-driven method. We then report the result of this comparison.

6.2.1. Ground-truth ontology construction method

In this survey, participants are asked to choose a relation for pair ⟨A, B⟩ from one of the following six options [13]:
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".

Each response is recorded as a participant preference for the chosen relation. Participants are only asked to compare information types that share at least one word.

Unlike an expert-based ground truth that assumes one or two viewpoints define truth, a population-based ground truth must reconcile different, even conflicting viewpoints. Diverse participant experiences can lead participants to perceive different information type senses and

to assign different semantic relations to the same pair: e.g., "mac" can be construed to mean either a MAC address for Ethernet-based routing, or a kind of computer sold by Apple Inc. In another example, the information type "email" can refer to three different senses: a service or program for sending messages; a message to be sent via the SMTP protocol; or a person's email address, which is the recipient address of an email message. Therefore, participants may conclude either that "email address" is a part of "email" or is equivalent to "email", which are both valid interpretations. To avoid excluding valid interpretations, we build a multi-viewpoint ground truth that accepts multiple, competing interpretations [13].

In this experimental design, *valid interpretations* for a pair are those 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 [13]. The expected response counts for a relation are based on how frequently participants choose that relation across all participant comparisons, which is the probability that a participant chooses a relation independent of the information types being compared. With this definition of valid interpretation, the multi-viewpoint ground truth is constructed as follows: for each surveyed pair, we add an axiom to the ground truth for a given relation, if the number of participant responses is greater than or equal to the expected Chi-square frequency for that pair and relation; except, if the number of unrelated responses exceeds the expected Chi-square frequency, then we do not add any axioms.

The above survey design is used to build two ground truths upon lexicon L_1 and L_3, which are used to conduct two experiments $E_{2.1}$ and $E_{2.2}$, respectively. Experiment $E_{2.1}$ aims to answer RQ2 by comparing the syntax-driven method with population preferences. Experiment $E_{2.2}$ serves as a replication in which the information types and domains vary, but wherein the standard of using population preferences is preserved. Thus, experiment $E_{2.2}$ on lexicon L_3 is also used to address RQ3 regarding method reliability.

6.2.2. Experiment $E_{2.1}$: Lexicon L_1

To address RQ2 and evaluate the syntax-driven method against population preferences for lexicon L_1, we published a population preference study and constructed a ground-truth ontology [34] using valid interpretations produced using survey design and analysis procedure described above. We refer to this ground-truth ontology as $KB_{2.1}$ throughout the paper.

Inferred relationships through the syntax-driven method. As mentioned in experiment $E_{1.1}$ (see Section 6.1), Lexicon L_1 contains 356 platform information types that are used to develop our context-free grammar (CFG) in Section 5.3. We follow the similar approach described in experiment $E_{1.1}$ Section 6.1 that yields 4593 relations between information types in L_1. The process of inferring candidate relations from lexicon L_1 is shown in Fig. 2. In this figure, L_1 candidate relations is an output artifact that we use as an input artifact in Fig. 5.

Ground-truth ontology. The survey is conducted using 2252 information type pairs obtained from L_1, wherein each pair consists of information types that share at least one word in the reduced version of lexicon L_1. These pairs are also previously used in $\widehat{KB_{1.1}}$ for experiment $E_{1.1}$. To identify relationships between information types in each pair, we recruit participants from AMT, where each participant had completed over 5000 HITs, had an approval rating of at least 97%, and was located within the United States [13]. Overall, we recruit 30 participants to provide one preferred relationship for each information type pair. This survey results in the ground-truth ontology $KB_{2.1}$ for this experiment.

Evaluation. We compare the relations inferred by the syntax-driven method with the relations in $KB_{2.1}$. An inferred relation is a true positive (TP), if it is logically entailed by the $KB_{2.1}$, otherwise, that relation is a false positive (FP). Overall, 979 inferred relations are logically entailed in $KB_{2.1}$. 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 $KB_{2.1}$. We only find two inferred relations as FPs. An unrelated information type pair in $KB_{2.1}$ is considered a true negative (TN), if we cannot match any inferred relations with those pairs. We find 805 pairs in $KB_{2.1}$ that are TNs. We count as false negatives (FN) all pairs in $KB_{2.1}$, which are not inferred by the method. We find 466 pairs in $KB_{2.1}$ that are not inferred by the method.

Table 11
Performance measures for Lexicon L_1 using population preferences.

Experiment	Method	Ground-Truth Ontology	TP	FP	TN	FN	Prec.	Rec.
$E_{2.1}$	Syntax-Driven method with CFG and Semantic Attachments	$KB_{2.1}$	979	2	805	466	0.997	0.677
$E_{2.1}$	26 Regular Expression Patterns	$KB_{2.1}$	781	3	878	590	0.996	0.569

We compute Precision= TP/(TP+FP) and Recall= TP/(TP+FN) for the syntax-driven method based on CFG and semantic attachments, presented in Table 11. In addition, we include results previously reported by Hosseini et al. in which an ontology was constructed from L_1 using 26 regular expressions over a shallow typology [13]. The new method outperforms the 26 regular expressions by decreasing the number of FNs and thus improving recall by 0.108.

6.2.3. Experiment $E_{2.2}$: Lexicon L_3

RQ3 asks about the method's reliability, which we address by conducting experiment $E_{2.2}$ using population preferences.

Inferred relationships through the syntax-driven method. For this experiment, we acquire lexicon L_3, that contains 1853 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) [14]. As shown in Fig. 5, in step 1, we pre-process 1853 information types in lexicon L_3 using the strategies mentioned in Section 5.1, yielding 1693 information types [34]. Step 2 involves manual effort for semantic tagging. During this step, two analysts (i.e., the first and second authors) individually assign tags to the reduced information types in L_3. The inter-rater agreement for the assigned tags is calculated using Fleiss' Kappa co-efficient. The comparison results in 518 disagreements with Kappa = 0.704. After reconciling the disagreements, Kappa is increased to 0.917 and we randomly select tag assignments from one of the analysts. In step 3, the tagged information types are parsed using the context-free grammar (CFG), yielding L_3-candidate relations as an output artifact of this process.

Ground-truth ontology. To address RQ3 on method reliability, we develop a ground truth for relations in L_3. Consistent with our prior experiments, we select information type pairs that share at least one word, yielding 1,466,328 pairs that are down-sampled using strata based on 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 a single stratum.

Phase B: Proportional stratified sampling is used to draw *at least* 2000 samples from all strata with the stratum size range 1–490. The wide range in stratum sizes implies unbalanced strata; e.g., a stratum that contains 1–3 pairs when divided by the total number of information type pairs yields zero, which would then be excluded from the sampled dataset. Therefore, we select all the pairs from stratum with size one to ensure complete coverage. For stratum of size two and three, we

Table 12
Performance measures for Lexicon L_3 using population preferences.

Experiment	Method	Ground-Truth Ontology	TP	FP	TN	FN	Prec.	Rec.
$E_{2.2}$	Syntax-Driven method with CFG and Semantic Attachments	$KB_{2.2}$	1686	3	438	156	0.998	0.915
$E_{2.2}$	26 Regular Expression Patterns	$KB_{2.2}$	952	9	748	574	0.990	0.623

randomly select one information type pair. For the remaining stratum with sizes greater than three, sample sizes are proportional to the stratum 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. In each stratum, we select the pairs that contain the least common lemmas with the bag-of-lemmas. 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.

Finally, 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. This final assurance increase the initial sample size of 2000 to the final size of 2283 samples from the original set of 1,466,328 phrase pairs. Our samples contain 1138 unique information types from Lexicon L_3.

We recruit 30 qualified AMT participants following the criteria mentioned in Section 6.1. These participants receive the same survey prompts described in $E_{2.1}$. Using the survey results, we construct a multi-viewpoint ground-truth ontology containing 2283 semantic relations [34]. We refer to this ontology as $KB_{2.2}$ throughout this paper.

Evaluation. The results of applying the syntax-driven method to the sampled information types from L_3 yields 21,745 inferred relations [34]. To compute precision and recall, we compare the inferred relations with the multi-view $KB_{2.2}$ using the same procedure described in experiment $E_{2.1}$. Overall, the method correctly identifies 1686 of the 2283 relations in $KB_{2.2}$. In comparison to $E_{2.1}$, the lexicon L_3 yields a higher recall by 0.238 with an equivalent precision. Overall, the method's precision remains high with few FPs, however, recall can vary by as much as 0.238 depending on the dataset. We also compare the inferred relations using 26 regular expressions [13] with $KB_{2.2}$. The performance measures in Table 12, suggest that our syntax-driven method reduces the number of FNs, thus improving recall by 0.292.

7. Limitations

In this section we discuss limitations of the work by analyzing false negatives. While the syntax-driven method affords high precision, the low recall presents a limitation of the approach. To understand this limitation, we open code the false negatives (FNs) for all four experiments (i.e., $E_{1.1}$, $E_{1.2}$, $E_{2.1}$, and $E_{2.2}$) and identify four explanations for this limitation. The four explanations are as follows.

(1) *Tacit Knowledge:* The method solely relies on syntax and infers semantic relations between information types that share at least one common word (e.g., "device identifier" and "identifier"). Consequently, relations between information types that rely on privacy policies' contextual semantics and experts' tacit knowledge are ignored. For example, the hypernymy relation between "mobile phone model" and "mobile device model" requires the knowledge that "phone" is a hyponym (kind) of "device". In another example, the hypernymy relation identified by non-experts 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.

(2) *Parse Ambiguity:* The syntax-driven method analyzes information types by grouping words from the right and left using the CFG and inherited variants in semantic attachments, respectively. For example, the experts identified a hypernymy relation between "mobile device unique device ID" and "mobile device ID". However, applying our method on the given information type "mobile device unique device ID" would not infer "mobile device ID" as a variant. In another example, an equivalence relation identified by non-experts between "device unique identifier" and "unique device identifier" would be inferred as two kinds of "device identifier", but not as equivalent concepts.

(3) *Role Suppression:* Experts and non-experts may ignore the roles regarding modifiers, events, and properties in information types, which results in relations that cannot be inferred by our method. For example, experts identified a hypernymy relation between "coarse location" and "actual location", ignoring "coarse" and "actual" that modify the term "location". In another example, experts identified a hypernymy relation between "device ID" and "access device", ignoring the role of "access" in "access device". Further, non-experts identified "actual location" and "approximate location" as equivalent information types in population preference studies.

(4) *Unjustifiable:* Some information type pairs and their relations in the ground truths of four experiments cannot be justified by our knowledge. For example, experts identified "contact list" as a synonym of "contact entry". In another example, "unique browser" is identified as a kind of "unique application number" by the experts. In another example, non-experts identified "general demographic information" as a kind of "general geographic information" and " mobile device type" is identified as a kind of "mobile device unique identifier".

Table 13 depicts the result of our analysis on the FN relationships that cannot be identified using the syntax-driven method for four experiments. This table shows the number of relations that fall in each coding category. To summarize, the majority of FNs in all four experiments belong to the "tacit knowledge" coding category, which shows the limitation of the syntax-driven method to infer relations that rely on privacy policy context, experts' domain knowledge, and humans' general knowledge of natural language.

8. Threats to validity

In this section we discuss threats to validity of the work.

Internal Validity- Internal validity concerns whether the inferences drawn from the experiments are valid. The relations inferred by the syntax-driven method depend on reliable labeling of information types by analysts (see Section 5.2). Changes in tags affect the performance of the method when compared to a ground truth.

To understand the effect of labels on the inferred relations by the syntax-driven method, we analyze the FNs for experiments $E_{1.1}$ and $E_{2.1}$ through a second-cycle coding of the "Tacit Knowledge" category (see Section 7). Recall that both $E_{1.1}$ and $E_{2.1}$ are organized using lexicon L_1 and the syntax-driven method is the result of grounded analysis on the role tagged information types in this lexicon. Through this analysis, we observed a potential explanation for why participants (i.e., experts and non-experts) prefer a relation that differs from our results. As an example, the terms in "application software" are tagged with the sequence *tt*, which is used to entail that "software" is part of an "application" by the syntax-driven method. However, we believe that participants recognize that "application software" is a single entity or thing (tag sequence *t*). We also believe this explanation applies to 20 information types in lexicon L_1. In summary, semantic ambiguity in tokenization and tagging can result in changes to the inferred relations, which is a shortcoming of the method. To minimize the effects of this internal threat, the information types in lexicons L_1, L_2, and L_3 are tagged by two analysts (the first and second authors). The reliability of

Table 13
False negative analysis.

Experiment	Ground truth	FNs total No.	Tacit knowledge	Parse ambiguity	Role suppression	Unjustifiable
$E_{1.1}$	$\widehat{KB_{1.1}}$	344	268	16	59	1
$E_{1.2}$	$\widehat{KB_{1.2}}$	44	33	5	0	0
$E_{2.1}$	$KB_{2.1}$	466	404	17	34	11
$E_{2.2}$	$KB_{2.2}$	156	128	6	18	4

the agreement between analysts when assigning the role tags is measured using Fleiss Kappa. Further, the ground-truth ontology $KB_{1.1}$ for experiment $E_{1.1}$ is constructed by two experts, i.e., the first and second authors of this paper. The ground-truth ontology $KB_{1.2}$ for experiment $E_{1.2}$ is a dataset previously published by our group [16]. The experts for $KB_{1.2}$ are the first author of this paper and the second author of [16]. The reliability of the agreement between experts when identifying semantic relationships between information types is measured using Fleiss Kappa. Finally, we acknowledge that our proposed method can be subject to researcher bias since the ground-truth ontologies in $E_{1.1}$ and $E_{1.2}$ are constructed by our research group.

External Validity- External validity concerns the extent to which the results generalize beyond the experimental setting. The syntax-driven method is constructed using lexicon L_1, which contains 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" [11]. The information types are extracted from collection data practices of 50 mobile app privacy policies [11,13]. We design experiments $E_{1.1}$ and $E_{2.1}$ to evaluate the method on lexicon L_1. To study generalizability beyond lexicon L_1 and reduce threats to external validity, we utilize lexicons L_2 and L_3 to evaluate and conduct experiments $E_{1.2}$ and $E_{2.2}$. Lexicon L_2 covers user-provided information from mobile apps in finance, health, and dating domains, whereas lexicon L_3 covers information types related to collection, usage, retention, and transfer data practices, extracted from web app privacy policies in shopping, telecommunication, social networks, employment, health, and news domains [14]. The syntax-driven method infers relations from L_2 with 0.800 precision and 0.840 recall in experiment $E_{1.2}$, when compared to information types that share at least one word in $\widehat{KB_{1.2}}$. In experiment $E_{2.2}$, the syntax-driven method results in 0.998 and 0.915 precision and recall, respectively, when compared to the population preferences for lexicon L_3. Experiments $E_{1.2}$ and $E_{2.2}$ suggests that the method generalizes best for general information types than for platform information types.

Reliability- Reliability indicates that the researcher's approach is consistent across different researchers and different projects [61]. To ensure the reliability of our research, we measure and report the inter-coder agreement in three ways [62]: (1) the Kappa for information type role tagging; (2) the Kappa measured for the expert-constructed ontologies; and (3) the Chi-square test used to threshold population preferences.

9. Discussion

Privacy policies contain ambiguous, general, and vague terminology, making requirements elicitation from data practices a challenging task. Our methodology provides requirements analysts with reusable ontologies that formalize semantic relations between ambiguous, general, and vague information types in privacy policies. However, such research poses fundamental challenges due to the problem's nature that requires systematic validation. In the following subsections, we provide brief responses to research questions, address the main challenges in this research, and provide lessons learnt.

9.1. Ontology construction methods in practice

To the best of our knowledge, our work is the first attempt to address ambiguity, generality, and vagueness by automatically inferring semantic relations, such as hypernymy, meronymy, and synonymy between information types in privacy policies. Further, we formalize the representation of these relationships into an ontology, which have been previously used to identify privacy violations in mobile applications [11,16]. Prior work to construct ontologies discovered seven heuristics that can be manually applied to infer semantic relations [10]. We build upon this method in Section 6.1.1 by constructing ground-truth ontologies $KB_{1.1}$ and $KB_{1.2}$ that are used in experiments $E_{1.1}$ and $E_{1.2}$. Consequently, the syntax-driven method described in this paper has only been shown to infer fragments of manually-constructed ontologies, in which the related information types share at least one word. Considering the manual and syntax-driven methods, we pose two questions for discussion. First, how do the two methods compare considering the time and manual labor required to construct an ontology? Secondly, what percentage of an ontology can be automatically generated using the proposed method?

To address the first question, we note that the manual method requires comparing paired information types by at least two analysts. Therefore, a complete analysis of n information types requires $\frac{n\times(n-1)}{2}$ comparisons. The average time for a pair comparison is reported at 11.72 seconds [13], which makes the task tedious and not feasible for large n. Further, the manual approach is susceptible to human error due to fatigue and gaps in analysts' domain knowledge [13]. In addition, as language use evolves, the ontology would need to be extended over time. Evidence from Bhatia et al. shows that between 23%–71% of information types in any new privacy policy will be previously unseen [63], which further motivates the need for a high-precision, semi-automated method to infer ontological relationships. Unlike the quadratic ($\mathcal{O}(n^2)$) number of paired comparisons required to identify relationships among information types, the syntax-driven method only requires a manual tagging step, which is linear ($\mathcal{O}(n)$) in the size of the lexicon. Furthermore, word tags can be reused across information type phrases that reuse words to further reduce the time needed to perform this step.

To address the second question, we first examine characteristics of the ground-truth ontology $KB_{1.1}$ in experiment $E_{1.1}$. Lexicon L_1 used in this experiment contains 356 information types, yielding $\frac{(356\times355)}{2}$ = 63,190 possible information type pairs and relationships. According to Table 6, 89% of pairs are identified as unrelated in $KB_{1.1}$. This percentage reflects that: (1) given all possible information type pair comparison, there are few relationships (high sparsity); and (2) the corresponding imbalance between related and unrelated pairs is intrinsic to this problem's nature. In addition, $KB_{1.1}$ contains 2015 positive relationships (i.e., direct hypernymy, synonymy, and meronymy). According to Table 7 for experiment $E_{1.1}$ and $KB_{1.1}$, the model identifies 1347 (TP) relationships from this positive space. Further, high precision of 91% suggests that a requirements analyst can trust the positive relationships (hypernymy, meronymy, and synonymy) inferred by the model. The problem of low recall (19%) arises from 5505 pairs falsely labeled as unrelated by the model. However, the model correctly identifies 60,315 unrelated pairs (TN). Based on the results in experiment $E_{1.1}$ using $KB_{1.1}$, we conclude that the model can identify most unrelated pairs; however, it cannot reduce the burden of manually checking the unrelated pairs for an analyst.

9.2. Real-world applications of ontologies

The use of ontologies are directly impactful to novel techniques in privacy-sensitive static and dynamic information flow tracing in software engineering [11,55,64]. However, due to technical challenges in information flow tracing, such approaches may not yet be broadly used in industry. That said, continued pressure by regulators and corporate compliance under privacy laws, including the E.U. General Data Protection Regulation, is driving the need to understand where companies collect and use sensitive data. A preliminary step in rationalizing corporate data flows, is classifying stored data by a broad category, which evidence suggests can easily reach into the thousands of distinct types [63]. Without explicit, scalable procedures to construct these ontologies and detect relationships between information type names, companies will produce compliance gaps, where rules for protecting data are applied unevenly due to misalignments arising from ambiguous and vague information type names. Our method for inferring semantic relations between information types can bolster these efforts and increase their viability by providing a general approach for ontology construction, thus yielding a strong indirect benefit to engineers and regulators.

9.3. Analysis of false negatives

Research questions **RQ1** and **RQ2** investigate the extent that the method can infer relations compared to relationships identified by experts and non-experts. To address these questions, we thoroughly analyze the relations that cannot be inferred by the method and are defined by experts or non-experts in ground truths. This category of relations are counted as false negatives (FNs) in the experiments. The analysis of FNs reveals that the majority of these relations are missed by the method due to the tacit knowledge required to infer these relations. Because the method is driven by word-level role labels or tags, the method is generalizable as the assignment of a role label to a word is dependent on the meaning of the word and is stable over time. Moreover, the small number of role labels improves reuse and requires less effort in labeling new words. However, as evidenced by analysis of FNs (see Section 7), these labels do not cover tacit knowledge signaled by the type phrases and encoded in expert experiences and non-expert understandings.

9.4. Method validity

Experiments $E_{1.1}$ and $E_{2.1}$ evaluate the syntax-driven method using lexicon L_1. However, the syntax-driven method is constructed upon this lexicon L_1. To address the validity of the method, we also design experiments $E_{1.2}$ and $E_{2.2}$ using lexicons L_2 and L_3. Lexicon L_2 covers user-provided information from mobile apps in finance, health, and dating domains, whereas lexicon L_3 covers information types related to collection, usage, retention, and transfer data practices, extracted from web app privacy policies in shopping, telecommunication, social networks, employment, health, and news domains [14]. The results from these two experiments suggest that the method generalizes best for general information types than for platform information types.

9.5. Method reliability on unseen information types

To evaluate the reliability of the syntax-driven method for unseen information types and address research question **RQ3**, we conduct two additional experiments using lexicons L_2 and L_3 containing information types extracted from various data practices of mobile and web-based apps' privacy policies in various domains. Our analysis reveals that the method can be reliably generalized for different information types in privacy policies with high precision and recall. Using our method to infer the semantic relations between information types reduces the ambiguity and improves the shared understanding between different stakeholders. Overall, 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.

9.6. Expert vs. non-expert evaluation

Comparing results of experiments $E_{1.1}$ and $E_{2.1}$, the non-expert population preferences in $E_{2.1}$ result in higher precision and recall by 0.15 and 0.19, respectively, in comparison with expert relations. We believe this difference in performance measures is due to the manual approach used by experts to construct the ontology. In the manual approach, experts are presented with all the information types in lexicon L_1 without common word restriction. Experts scan the alphabetical list of the information types, looking for types with which they can create hypernymy, meronymy, or synonymy relations. Experts can search by keyword, and largely rely on their memory of information types they have seen as they discover relationships. For our evaluation purposes, we sample a set of relations between information type pairs that share at least one common word from the manually constructed ontology by experts. However, to construct the population preferences, we first select a set of information type pairs that share at least one common word from lexicon L_1 and finally this set is presented to non-experts during the study. As a result, this set is more intuitive, making it simpler for participants to identify the relationships between information types presented to them without scanning an alphabetical list and trying to retain information types they have encountered in the list. Therefore, using non-expert views as ground truth results in higher precision and recall when compared to expert views.

10. Conclusion

Privacy policies are legal documents containing application data practices. These documents are well-established sources of requirements in software engineering. However, privacy policies are written in natural language, thus subject to ambiguity and abstraction. Eliciting requirements from privacy policies is a challenging task as these ambiguities can result in more than one interpretation of a given term. In this paper, we focus on the role of hypernyms, meronyms, and synonyms and their formal relationships among terminology in privacy policies. We propose a novel, automated syntax-driven semantic analysis method for constructing partial ontologies to formalize these relationships. Formal ontologies can be used as knowledge bases by requirements analysts for resolving conflicting interpretations of ambiguous terminology. The syntax-driven method infers semantic relations between information types in privacy policies and their morphological variants based on a context-free grammar and its semantic attachments. This method is constructed based on grounded analysis of lexicon L_1 containing platform information types extracted from collection data practices of 50 mobile app privacy policies.

The current paper extends a previous conference paper [21]. In the conference version, we only evaluated our method against information types that share at least one common word in lexicons L_1 and L_3 using population preference surveys. The current paper extends the evaluation by introducing the expert-constructed ontologies as ground truths (i.e., $KB_{1.1}$ and $KB_{1.2}$) in experiments $E_{1.1}$ and $E_{1.2}$. However, our syntax-driven method fails to infer the relations in $KB_{1.1}$ and $KB_{1.2}$ that are between information types that share no common word. We consider such relationships out of scope for the technical contribution of this paper. Therefore, we introduce $\widehat{KB_{1.1}}$ and $\widehat{KB_{1.2}}$ (i.e., subsets of $KB_{1.1}$ and $KB_{1.2}$) as ground truths. In summary, experiment $E_{1.1}$ results in 84% precision and 48% recall when compared to expert views in $\widehat{KB_{1.1}}$. Further, experiment $E_{1.2}$ results in 80% precision and 84% recall when compared to expert views in $\widehat{KB_{1.2}}$. To show the broad image of the problem, we also report the results of our syntax-driven method compared to $KB_{1.1}$ and $KB_{1.2}$. We emphasize that our method is not capable of identifying the relations in this broad space. We foresee using word embeddings trained using Word2Vec and language models such as BERT that can address this shortcoming for our future work.

CRediT authorship contribution statement

Mitra Bokaei Hosseini: Conceptualization, Methodology, Software, Validation, Formal analysis, Data curation, Writing - original draft, Visualization. **Travis D. Breaux:** Conceptualization, Software, Validation, Formal analysis, Investigation, Writing - review & editing, Supervision, Funding acquisition. **Rocky Slavin:** Writing - review & editing. **Jianwei Niu:** Methodology, Formal analysis, Resources, Funding acquisition, Supervision. **Xiaoyin Wang:** Methodology, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We thank Xue Qin for her participation as an expert in experiment $E_{1.2}$. This research was supported by NSF, United State awards #1736209, #1748109, #2007718, #1453139, and #1948244.

References

[1] K. Harris, Privacy on the Go: Recommendations for the Mobile Ecosystem, California Department of Justice, 2013.
[2] A.I. Anton, J.B. Earp, A requirements taxonomy for reducing web site privacy vulnerabilities, Requir. Eng. 9 (3) (2004) 169–185.
[3] T.D. Breaux, H. Hibshi, A. Rao, Eddy, a formal language for specifying and analyzing data flow specifications for conflicting privacy requirements, Requir. Eng. 19 (3) (2014) 281–307.
[4] A.K. Massey, J. Eisenstein, A.I. Antón, P.P. Swire, Automated text mining for requirements analysis of policy documents, in: 2013 21st IEEE International Requirements Engineering Conference, RE, IEEE, 2013, pp. 4–13.
[5] A.K. Massey, E. Holtgrefe, S. Ghanavati, Modeling regulatory ambiguities for requirements analysis, in: International Conference on Conceptual Modeling, Springer, 2017, pp. 231–238.
[6] E. Kamsties, B. Peach, Taming ambiguity in natural language requirements, in: Proceedings of the Thirteenth International Conference on Software and Systems Engineering and Applications, 2000.
[7] J.R. Reidenberg, J. Bhatia, T.D. Breaux, T.B. Norton, Ambiguity in privacy policies and the impact of regulation, J. Legal Stud. 45 (S2) (2016) S163–S190.
[8] D.M. Berry, E. Kamsties, Ambiguity in requirements specification, in: Perspectives on Software Requirements, Springer, 2004, pp. 7–44.
[9] A.K. Massey, R.L. Rutledge, A.I. Antón, P.P. Swire, Identifying and classifying ambiguity for regulatory requirements, in: 2014 IEEE 22nd International Requirements Engineering Conference, RE, IEEE, 2014, pp. 83–92.
[10] M.B. Hosseini, S. Wadkar, T.D. Breaux, J. Niu, Lexical similarity of information type hypernyms, meronyms and synonyms in privacy policies, in: AAAI Fall Symposium, 2016.
[11] R. Slavin, X. Wang, M.B. Hosseini, J. Hester, R. Krishnan, J. Bhatia, T.D. Breaux, J. Niu, Toward a framework for detecting privacy policy violations in Android application code, in: Proceedings of the 38th International Conference on Software Engineering, 2016, pp. 25–36.
[12] J. Bhatia, T.D. Breaux, J.R. Reidenberg, T.B. Norton, A theory of vagueness and privacy risk perception, in: 2016 IEEE 24th International Requirements Engineering Conference, RE, IEEE, 2016, pp. 26–35.
[13] M.B. Hosseini, T.D. Breaux, J. Niu, Inferring ontology fragments from semantic role typing of lexical variants, in: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, 2018, pp. 39–56.
[14] M.C. Evans, J. Bhatia, S. Wadkar, T.D. Breaux, An evaluation of constituency-based hyponymy extraction from privacy policies, in: 2017 IEEE 25th International Requirements Engineering Conference, RE, IEEE, 2017, pp. 312–321.
[15] M. Jackson, Problems and requirements, in: Proceedings of 1995 IEEE International Symposium on Requirements Engineering, RE'95, IEEE, 1995, pp. 2–8.
[16] X. Wang, X. Qin, M.B. Hosseini, R. Slavin, T.D. Breaux, J. Niu, Guileak: Tracing privacy policy claims on user input data for Android applications, in: Proceedings of the 40th International Conference on Software Engineering, 2018, pp. 37–47.
[17] K.K. Breitman, J.C.S. do Prado Leite, Ontology as a requirements engineering product, in: Proceedings. 11th IEEE International Requirements Engineering Conference (RE), 2003, IEEE, 2003, pp. 309–319.
[18] G. Frege, Über begriff und gegenstand, OR Reisland, 1892.
[19] T.M. Janssen, B.H. Partee, Compositionality, in: Handbook of Logic and Language, Elsevier, 1997, pp. 417–473.
[20] E. Bach, An extension of classical transformational grammar, 1976.
[21] M.B. Hosseini, R. Slavin, T. Breaux, X. Wang, J. Niu, Disambiguating requirements through syntax-driven semantic analysis of information types, in: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, 2020, pp. 97–115.
[22] T.D. Breaux, D.L. Baumer, Legally "reasonable" security requirements: A 10-year FTC retrospective, Comput. Secur. 30 (4) (2011) 178–193.
[23] FTC, FTC's $5 billion Facebook settlement: Record-breaking and history-making, 2019.
[24] T.D. Breaux, A.I. Antón, E.H. Spafford, A distributed requirements management framework for legal compliance and accountability, Comput. Secur. 28 (1–2) (2009) 8–17.
[25] G. Petronella, Analyzing privacy of Android applications, Italy, 2014.
[26] S. Zimmeck, Z. Wang, L. Zou, R. Iyengar, B. Liu, F. Schaub, S. Wilson, N. Sadeh, S.M. Bellovin, J. Reidenberg, Automated analysis of privacy requirements for mobile apps, in: NDSS, 2017.
[27] J. Bhatia, T.D. Breaux, Towards an information type lexicon for privacy policies, in: RELAW, IEEE, 2015, pp. 19–24.
[28] R. Slavin, X. Wang, M.B. Hosseini, J. Hester, R. Krishnan, J. Bhatia, T.D. Breaux, J. Niu, PVDetector: a detector of privacy-policy violations for Android apps, in: 2016 IEEE/ACM International Conference on Mobile Software Engineering and Systems, MOBILESoft, IEEE, 2016, pp. 299–300.
[29] P. Slovic, The construction of preference, Amer. Psychol. 50 (5) (1995) 364.
[30] D. Jurafsky, J.H. Martin, Speech and Language Processing, Vol. 3, Pearson London, 2014.
[31] F. Baader, D. Calvanese, D. McGuinness, P. Patel-Schneider, D. Nardi, et al., The Description Logic Handbook: Theory, Implementation and Applications, Cambridge University Press, 2003.
[32] F. De Saussure, R. Harris, Course in General Linguistics. (Open Court Classics), Open Court, Chicago and La Salle, 1998.
[33] C. Hookway, Peirce-Arg Philosophers, Routledge, 2010.
[34] M. Bokaei Hosseini, T. Breaux, Privacy Policy Ontology, Mendeley Data, 2020, http://dx.doi.org/10.17632/46r2vd7jnv.1.
[35] IEEE recommended practice for software requirements specifications, in: IEEE Computer Society. Software Engineering Standards Committee and IEEE-SA Standards Board, Vol. 830, IEEE, 1998.
[36] R. Harwell, E. Aslaksen, R. Mengot, I. Hooks, K. Ptack, What is a requirement? in: INCOSE International Symposium, Vol. 3, No. 1, Wiley Online Library, 1993, pp. 17–24.
[37] G. Lami, S. Gnesi, F. Fabbrini, M. Fusani, G. Trentanni, An Automatic Tool for the Analysis of Natural Language Requirements, Informe técnico, CNR Information Science and Technology Institute, Pisa, Italia, Setiembre, 2004.
[38] B. Gleich, O. Creighton, L. Kof, Ambiguity detection: Towards a tool explaining ambiguity sources, in: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, 2010, pp. 218–232.
[39] A. Nigam, N. Arya, B. Nigam, D. Jain, Tool for automatic discovery of ambiguity in requirements, Int. J. Comput. Sci. Issues 9 (5) (2012) 350.
[40] L. Mich, R. Garigliano, et al., Ambiguity measures in requirement engineering, in: International Conference on Software Theory and Practice, ICS, 2000.
[41] N. Kiyavitskaya, N. Zeni, L. Mich, D.M. Berry, Requirements for tools for ambiguity identification and measurement in natural language requirements specifications, Requir. Eng. 13 (3) (2008) 207–239.
[42] H. Yang, A. De Roeck, V. Gervasi, A. Willis, B. Nuseibeh, Extending nocuous ambiguity analysis for anaphora in natural language requirements, in: 2010 18th IEEE International Requirements Engineering Conference, RE, IEEE, 2010, pp. 25–34.
[43] H. Yang, A. De Roeck, V. Gervasi, A. Willis, B. Nuseibeh, Analysing anaphoric ambiguity in natural language requirements, Requir. Eng. 16 (3) (2011) 163.
[44] A. Ferrari, S. Gnesi, Using collective intelligence to detect pragmatic ambiguities, in: 2012 20th IEEE International Requirements Engineering Conference, RE, IEEE, 2012, pp. 191–200.
[45] A. Ferrari, P. Spoletini, S. Gnesi, Ambiguity and tacit knowledge in requirements elicitation interviews, Requir. Eng. 21 (3) (2016) 333–355.
[46] V. Gervasi, D. Zowghi, On the role of ambiguity in RE, in: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, 2010, pp. 248–254.
[47] S. Boyd, D. Zowghi, V. Gervasi, Optimal-constraint lexicons for requirements specifications, in: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, 2007, pp. 203–217.
[48] G.A. Miller, WordNet: a lexical database for English, Commun. ACM 38 (11) (1995) 39–41.
[49] D. Fensel, D. McGuiness, E. Schulten, W.K. Ng, G.P. Lim, G. Yan, Ontologies and electronic commerce, IEEE Intell. Syst. 16 (1) (2001) 8–14.

[50] L. Zhao, W. Alhoshan, A. Ferrari, K.J. Letsholo, M.A. Ajagbe, E. Chioasca, R.T. Batista-Navarro, Natural language processing (NLP) for requirements engineering: A systematic mapping study, 2020, arXiv preprint arXiv:2004.01099.
[51] D. Dermeval, J. Vilela, I.I. Bittencourt, J. Castro, S. Isotani, P. Brito, A. Silva, Applications of ontologies in requirements engineering: a systematic review of the literature, Requir. Eng. 21 (4) (2016) 405–437.
[52] D.G. Gordon, T.D. Breaux, Reconciling multi-jurisdictional legal requirements: A case study in requirements water marking, in: 2012 20th IEEE International Requirements Engineering Conference, RE, IEEE, 2012, pp. 91–100.
[53] A. Oltramari, D. Piraviperumal, F. Schaub, S. Wilson, S. Cherivirala, T.B. Norton, N.C. Russell, P. Story, J. Reidenberg, N. Sadeh, PrivOnto: A semantic framework for the analysis of privacy policies, Semant. Web 9 (2) (2018) 185–203.
[54] L. Humphreys, G. Boella, L. van der Torre, L. Robaldo, L. Di Caro, S. Ghanavati, R. Muthuri, Populating legal ontologies using semantic role labeling, Artif. Intell. Law (2020) 1–41.
[55] H. Harkous, K. Fawaz, R. Lebret, F. Schaub, K.G. Shin, K. Aberer, Polisis: Automated analysis and presentation of privacy policies using deep learning, in: 27th USENIX Security Symposium, 2018, pp. 531–548.
[56] S. Wilson, F. Schaub, A.A. Dara, F. Liu, S. Cherivirala, P.G. Leon, M.S. Andersen, S. Zimmeck, K.M. Sathyendra, N.C. Russell, et al., The creation and analysis of a website privacy policy corpus, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vol. 1, 2016, pp. 1330–1340.
[57] J. Corbin, A. Strauss, et al., Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory, Sage, Thousand Oaks, CA, 2008.
[58] J. Saldaña, The Coding Manual for Qualitative Researchers, Sage, 2015.
[59] H. Barendregt, The Lambda Calculus, its Syntax and Semantics, Elsevier Science Publishers, 1985.
[60] J.L. Fleiss, Measuring nominal scale agreement among many raters, Psychol. Bull. 76 (5) (1971) 378.
[61] G.R. Gibbs, Analyzing Qualitative Data, Vol. 6, Sage, 2018.
[62] J.W. Creswell, J.D. Creswell, Research Design: Qualitative, Quantitative, and Mixed Methods Approaches, Sage Publications, 2017.
[63] J. Bhatia, T.D. Breaux, F. Schaub, Mining privacy goals from privacy policies using hybridized task recomposition, ACM Trans. Softw. Eng. Methodol. 25 (3) (2016) 1–24.
[64] S. Zimmeck, S.M. Bellovin, Privee: An architecture for automatically analyzing web privacy policies, in: 23rd USENIX Security Symposium, 2014, pp. 1–16.