INFORMATION RETRIEVAL AND SEMANTIC INFERENCE FROM NATURAL LANGUAGE PRIVACY POLICIES

by

MITRA BOKAEI HOSSEINI, M.Sc.

DISSERTATION
Presented to the Graduate Faculty of
The University of Texas at San Antonio
In Partial Fulfillment
Of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT SAN ANTONIO
College of Sciences
Department of Computer Science
May 2019

INFORMATION RETRIEVAL AND SEMANTIC INFERENCE FROM NATURAL LANGUAGE PRIVACY POLICIES

Mitra Bokaei Hosseini
The University of Texas at San Antonio, 2019

Supervising Professor: Jianwei Niu, Ph.D.

Several state laws, along with app markets, such as Apple's App Store and Google Play, require app developers to provide users with legal privacy notices (privacy policy) containing critical requirements that inform users about what kinds of personal information is collected, how the data is used, and with whom the data is shared. Because privacy policies consist of legal terms often written by a legal team without rigorous insight into the app source code, and because the policy and app code can change independently, privacy policies become misaligned with the actual data practices. In addition to misinforming users, such inconsistencies between policies and data practices can have legal repercussions. The goal of this work is to capture and formalize the semantics of natural language privacy policies into a knowledge base that can actuate (1) transparent software implementation; and (2) shared understanding between policy authors, app developers, and regulators. Constructing an empirically valid knowledge base (i.e., privacy policy ontology) is a challenging task since it should be both scalable and consistent with multi-user interpretations.

This work focuses on formal representation of privacy policy semantics by applying grounded theory, natural language patterns, and neural networks on terminology of privacy policies. Further, the application of formal ontologies in privacy misalignment detection frameworks is discussed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1: INTRODUCTION

Mobile and web applications (apps) offer users innovated ways to access and share data. According to the PEW Research Center, 77% of U.S. adults own a smartphone and utilize mobile devices for activities, such as looking for jobs, finding dates, or reading books [56]. To provide users with these services, mobile apps collect various categories of personal information, such as contact information, photos, and real-time location. This information collection exposes users to potential privacy risks. To inform users with data practices, app developing companies publicly publish privacy notices, also called privacy policies. These polices are one of the major ways developers communicate data requirements to users and inform stakeholders about what kinds of personal information is collected, how the data is used, and with whom the data is shared [1, 33]. Further, various U.S. and E.U. laws [1] [2] require app developing companies to post their privacy polices and notify users with any changes in their data practices.

Privacy policies pose a technical challenge in requirements analysis and traceability. Being comprehensive, these policies often fail to provide a detailed description of companies' practices and mainly focus on abstract information types. Apart from abstraction, stakeholders use different words for the same domain concept, which reduces shared understanding of the subject and leads to a misalignment among app developers' intention, and policy authors, and regulators expectation [11]. Abstraction and variability in concept representation commonly found in privacy policies are that of hypernymy, which occurs when a more abstract information type is used instead of a more specific information type (e.g., "device information" instead of "IP address") [6]. Hypernymy permits multiple interpretations of words and phrases, which leads to inconsistency in tracing the requirements.

As an example, we compare two different versions of Adobe's privacy policy. The following is a snippet from Adobe's privacy policy last updated in December 8, 2014.

[1]CAL. CIVIL CODE §1798.140(c)
[2]https://gdpr-info.eu/

When you activate your Adobe product, we collect certain **information about your device**, the Adobe product, and your product serial number (learn more: http://www.adobe.com/products/activation/).

This statement mentions the collection of the abstract information type “information about your device” that can be interpreted in various ways, yielding ambiguity in requirements analysis and traceability. For example, since a mobile device is a kind of device, a requirements analyst can infer that the statement also implies the collection of “mobile device information.” Further, some devices have identifiers, such as a serial number, or IP address, thus Adobe can also collect device identifiers because device identifiers are a meronym or part of a device. Finally, the analyst can infer that the statement applies to mobile device identifier by reasoning over both the hypernymy and meronymy relationships, together. These interpretations are recognized by a person, such as a requirements analyst or a developer, and are matched with phenomena in the world based on their experience and background knowledge.

We now illustrate the recent version of Adobe’s privacy policy (last updated May, 3 2018[3]):

When you activate your Adobe app or when you install updates to the app, we collect **information about: your device** (**manufacturer**, **model**, **IP address**); the Adobe app (version, date of activation, successful and unsuccessful updates); your product serial number (where this is required before you can start using your product).

Within this recent update, we notice specific kinds of device information are listed in the privacy policy. Therefore, the requirements analyst can infer “device information” as an abstract or general term that can be used to describe “manufacturer,” “model,” and “IP address” as specific kinds of “device information.” Thus, collection of information when activating or installing updates also applies to these specific kinds. This snippet provides the stakeholders with more understanding of the Adobe’s data practices. Based on the specific terms, requirements analyst can construct a knowledge base that clarifies specific kinds of each abstract information type, which

[3]https://www.adobe.com/privacy/policy.html

can be used for requirements analysis and traceability in mobile apps.

To ensure data transparency and compliance, methods have been proposed to analyze privacy policies and data practices. For example, Breaux et al. formalized data requirements in privacy policies using Description Logic [10], which can then be used to automatically detect conflicting requirements [15] and to trace data flows across policies of interacting services [15]. Tracing privacy requirements across policies can enhance developers' understanding of third-parties' data practices and comply with legal requirements, such as General Data Protection Regulation (GDPR), Articles 13.1 and 14.12. Furthermore, others have proposed techniques to trace data requirements from privacy policies to app code using lookup tables, platform permissions, and information flow analysis [53, 74]. These methods depend on a correct lexicon of information types that is generally compiled manually. These information types include phrases, such as device ID, IP address, and location. Moreover, the phrases can overlap in meaning leading to false requirements formalization, e.g., the phrase "WiFi signal strength" can be construed to be a type of location information. Therefore, identifying data flows and trace links between natural language privacy policies and app code cannot be achieved without addressing abstraction and variability within privacy policy concept representation.

To overcome the abstraction and variability in concept representation, this thesis aims to capture and formalize the semantics of natural language privacy policies into a knowledge base, called ontology. Ontology is an arrangement of terminology into a hierarchy using semantic relations, such as ubclass/superclass, part/whole, and synonymy, among other categories. This formal representation of concepts and their interpretations can help requirements analysts, policy authors, app developers, and regulators to consistently check how data practice descriptions relate to one another and to identify unintended interpretations. Further, application of this knowledge base can actuate transparent software implementation for users.

There are two major challenges to this work. First, designing and evaluating solutions for this problems requires datasets in the mobile app privacy policy domain. Second, the proposed solutions should be generalizable and scalable considering both the growing number of apps in the market and the evolving privacy-related regulations.

In the next section, we discuss in detail our proposed methods to construct privacy policy ontology.

1.1 Proposed Methods for Constructing Privacy Policy Ontology

Abstraction and variability in concept representation sap companies' ability to consistently compare their privacy policies with (1) their actual data practices (2) and privacy notices of their third parties in a multi-tier service composition framework. This thesis studies approaches to construct information type ontologies on privacy policies. We believe ontologies as knowledge bases can provide share understanding between multiple stakeholders, such as requirements analysts, policy authors, app developers, regulators, and users. Further, ontologies enable requirements analysts to model data practices and reason the sources of inconsistencies and non-compliance between multiple software artifacts. In addition to misinforming users, such inconsistencies can have legal repercussions for app developing companies. To this end, we discuss three main methods to construct privacy policy ontologies.

As an initial attempt to construct a privacy-related information type ontology, we applied content analysis, which is a qualitative research method for annotating text to identify words and phrases that embody the meaning of special codes [59], and grounded theory [21] to discover seven heuristics for manually classifying information types into a formal ontology.

We evaluated these heuristics in a study of 50 mobile app privacy policies(see Section 5.4). We extracted information types related to automatic data collection practices from these 50 policies, yielding a platform information lexicon containing 356 information types discussed in Section 4. In the next step, two analysts compared two information types in the lexicon and assigned a semantic relationship based on the discovered heuristics. We compared the relationships assigned to the pairs by each analysts, calculating the inter-rater reliability and reconciling the differences to achieve high agreement for assignments. Finally, we formalized the relationships between pairs in an ontology using Description Logic.

In another study, we extracted user-provided information from data collection practices in privacy policies of 20 apps in health, finance, and dating domains. We applied a similar approach to construct three different ontologies for these domains.

Finally, we applied these ontologies in privacy violation detection framework to identify privacy violations (i.e., instances where privacy polices are inconsistent with app code data practices). The results from these studies are summarized in Sections 5.6.1 and 5.6.2.

As a motivating example, consider a policy snippet of app A: "We collect information, including device information and device ID." For constructing an ontology, we first extract "device information" and "device ID." Using heuristics, we infer subclass relationship between these two information types, which is added to the ontology. We can apply this ontology to check the policy and code alignment in app B, which collects "device ID" and discloses collection of "device information" in its policy. Using the ontology, we can acknowledge the company's disclosure of the information collection. Yet, raise a warning on the use of a phrase that is too general, which can potentially mislead the user or disregard policy regulation regulations.

This manual approach to construct ontologies requires at least two analysts to perform paired comparison of all information types. The required effort for this task is quadratic $m \times n \times (n-1)/2$, where n is the number of information types in the lexicon, and m is the amount of time required to assign a relationship to a pair, estimated at 20 seconds []. In brief, while the approach validates the concept, it requires improvements to address several limitations, including time consumption, scalability, and human error.

To improve our ontology construction approach, we developed a semi-automated syntax-based method that employs a shallow typology and regular expression rules to categorize individual words in information types and parse them for inferring semantic relations, such as hypernymy. We discovered these patterns and rules to represent and parse the information types through grounded analysis of information types extracted from 50 privacy policies. In this approach analysts assign semantic role types to the information type phrases, and then a set of regular expression rules are automatically applied to yield a subset of all possible relations.

To improve the semantic relations inferred using these initial set of regular expression rules, we established a ground truth using crowdsourcing by asking human subjects to perform the more time-consuming task of comparing information types in the lexicon. We then compared the results of the rules against these human interpretations, which led to identifying additional rules. Finally, we evaluated the extended set of rules on 109 unique information types extracted from six privacy policies, and human subject surveys to measure the correctness of the results produced by the rules (see Section 6.3.3).

To formalize the discovered patterns and regular expression rules, we utilize rule-to-rule hypothesis [4] that augments production rules in a context-free grammar (CFG) with semantic attachments. To this end, we introduced a CFG to present and parse information types in a privacy policy lexicon. Under a generative treatment of morphology, the CFG produces variants of a given information type phrase, which are necessary to build an ontology and overcome the abstraction problem in natural text. Finally, we developed a tool that augments CFG with semantic attachments to infer relations. For example, this formalization can derive "IP address" as a morphological variant from "device IP address," and infer part-whole relationship between them. The semantic attachments are formally presented using λ-calculus. Detailed description of this formalization is discussed in Chapter 7.

We evaluate this method as follows using two information type lexicon containing 356 platform-related information types [37, 60] and 1,853 general information types extracted from 30 policies [23]. We compare the inferred relations from applying CFG and semantic attachments on these lexicon with human classification results obtained from preference studies, in which crowd workers select the best semantic relation between two information types based on their background knowledge and experience. Despite high performance measures recorded for these studies, our method fails to infer semantic relations between some information types, such as "mobile device" and "iPhone," that don't share any common words and require additional embedded tacit knowledge for relation inference.

To address this limitation, we developed an empirical method to learn and construct a formal ontology from a set of information type pairs contained in a lexicon. This method applies a novel neural network classifier to predict semantic relationships between information type pairs. We also present a novel method to sample from an existing ontology to create training and testing sets that account for dependencies among concepts and formal ontological relations. Finally, we present the evaluation of our relationship classifier on a privacy policy ontology. The details for this proposed method and evaluation is presented in Chapter 8.

In future, we envision augmenting the syntax-based method with the neural network classifier, hence enhancing the relation inference between information type pairs relying on both syntax features and tacit knowledge. Based on our current evaluation, both methods only infer fragments of an ontology. Therefore, further empirical validation is required to identify the reachability of concepts in the ontology fragments generated through the augmented approach. We foresee cases where both methods fail to infer semantic relations and therefore, requirements analyst should be involved to infer additional semantic relations between information types. Initial evaluation is required to identify the amount of effort required for a requirements analyst to manually identify these cases.

Our work has broader impact on data flow and requirements traceability domains. We plan to integrate our work with data flow and requirements compliance detection tools that rely on precise description of information types. For example, Breaux et al. identifies data flow traces across privacy policies of a multi-tier service systems [12]. Further, Slavin et al. [60] and Wang et al. [69] investigate privacy requirements traceability in Android apps. A new policy introduces between 11-36 new information types that are not encountered in the existing lexicon [6]. Considering an average of 6,140 Android apps being released through Google Play Store everyday[4], there is a need for an automated approach that extends an existing ontology as a knowledge base on privacy policies. We envision extending the existing ontologies used in data flow and privacy violation detection tools and compare the results with the published conflicting requirements and violations in previous research [60, 69].

[4]https://www.statista.com/statistics/276703/android-app-releases-worldwide/

Further, we foresee defining the ontology construction problem as a paraphrasing task in natural language processing domain [50], where the goal is to predict a paraphrase template for a given information type presented using a noun compound.

In summary, this thesis aims at developing novel methods and tools to address abstraction in natural language privacy requirements by comprehending the semantic knowledge and formal representation of this knowledge as a reusable artifact.

1.2 Thesis Statement

Abstraction and variability in concept representation undermines companies' ability to successfully align their privacy policies with (1) their actual data practices (2) and privacy notices of their third parties in a multi-tier service composition framework. In addition to misinforming users, such inconsistencies between policies and data practices can have legal repercussions for app developing companies. Formal ontologies on privacy policies can mitigate the abstraction effects and help policy authors, app developers, and regulators to consistently check how data practice descriptions relate to one another and identify unintended interpretations. This thesis studies approaches to construct information type ontologies on privacy policies using (1) heuristics derived from grounded analysis of privacy policies; (2) regular expression rules; and (2) neural network classifier. The outcome of this thesis is privacy policy ontologies that can be used to identify trace links between software artifacts to achieve transparent software products.

The background and related work are presented in Chapters 2 and 3, respectively. In Chapter 4, the details about constructing information type lexicon from privacy policies are explained. In Chapter 5, we discuss the details of manual ontology construction using information type lexicon. Chapter 6 includes the details for the syntax-based method using regular expression rules for constructing a formal ontology. The details about formalizing the patterns and rules using rule-to-rule hypothesis to infer semantic relations are presented in Chapter 7. In Chapter 8, we discuss the neural network classifier to predict semantic relations for an information type pair. And finally, Chapters 9 and 10 present the future work and conclusion of this thesis.

Chapter 2: BACKGROUND

In this section, I introduce necessary background information that is required for understanding this work.

2.1 Semantic Relations

This section provide definitions for some of the relations that hold among natural language phrases.

Hypernymy: a relationship between two noun phrases, wherein one phrase is superordinate to the other term. The superordinate phrase is called the hypernym, and the subordinate phrase is called the hyponym. In English, one can say the hyponym "is a kind of" hypernym. For example, "device identifier" is a kind of "device information."

Meronymy: a semantic relation between two known phrases, wherein one phrase is part of the other term as a whole. The part is called meronym, and the whole containing the part is called the holonym terms. In English, one can say the meronym "is a part of" holonym. For example, "device MAC address" is a part of "device."

Synonymy: a relationship between two noun phrases with a similar meaning. For example, "IP address" is a synonym of "Internet protocol address."

Lexicon- a collection of phrases or concept names that may be used in an ontology.

Ontology- An arrangement of terms into a graph in which terms are connected via edges corresponding to semantic relations, such as hypernymy and synonymy, among others [41].

2.2 Description Logic Ontologies

Description Logic (DL) ontologies enable automated reasoning, including the ability to infer which concepts subsume or are equivalent to other concepts in the ontology. We chose tFL_0, a sublanguage of the Attribute Language (AL) in Description Logic (DL), which is PSPACE-complete for concept satisfiability and concept subsumption. A DL knowledge base KB is comprised of two components, the TBox and the ABox [3]. 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. The platform ontology knowledge base KB only contains terminology, which we call the TBox T.

The semantics of 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 set $C^{\mathcal{I}} \subseteq \ Delta^{\mathcal{I}}$. The TBox T also contains terminological axioms that relate concepts to each other in the form of subsumption and equivalence, which we use to formalize hypernymy and synonymy, respectively. A concept C is subsumed by a concept D, written $T \models C \sqsubseteq D$, if $C^{\mathcal{I}} \sqsubseteq D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$ that satisfy the TBox T. The concept C is equivalent to a concept D, written $T \models C \equiv D$, if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$ that satisfy the TBox T. Axioms of the first kind ($C \sqsubseteq D$) are called inclusions, where axioms of the second kind ($C \equiv D$) are called equalities [3]. Note that the equalities $C \equiv D$ can be rewritten as two inclusion axioms $C \sqsubseteq D$ and $D \sqsubseteq C$ [63]. For meronymy, we define a part-whole relation that maps parts to wholes as follows: a part concept C that has a whole concept D, such that $T \models C \sqsubseteq (partOfD)$. We express the DL ontology using the Web Ontology Language[1] (OWL) version 2 DL and the HermiT[2] OWL reasoner.

2.3 Andorid API

Applications can interact with underlying Android system using a framework API provided by the Android platform. The framework API contains set of packages, classes, and methods. The Android 4.2 framework is comprised of about 110,000 methods, some of which are specifically used to retrieve, insert, update, or delete sensor data through the Android OS [57]. The use of an API increases the level of security by not allowing apps to have direct access to all sensor data by default.

Before an app can access specific methods from the API, the required permissions must be requested by the app through a manifest file. An app's manifest file enumerates the app's required

[1]https://www.w3.org/TR/owl-guide/
[2]http://www.hermit-reasoner.com/

permissions and is described to users when installing an app as well as on the app's download page on the Google Play store. Thus, there is a direct relationship between the permissions granted to an application by a user at installation time and eligible API method calls in the application source code.

Morphological Variant: information type phrases are frequently variants of a common lexeme, e.g., "device" is a morphological variant of "mobile device."

In the definitions above, we assume that noun phrases expressed in text have a corresponding concept and that the text describes one name for the concept. This relationship between the phrase and concept is also arbitrary, as noted by Saussure in his theory of the signifier, which is the symbol that represents a meaning, and the signified, which is the concept or meaning denoted by the symbol [22]. Peirce defines a similar relationship between sign-vehicles and objects, respectively [36].

2.4 Context Free Grammar

A context-free grammar G is a quadruple $G =< N, T, R, S >$, where N is a final set of non-terminal symbols; T is a finite set of terminal symbols; R is a finite set of productions; and $S \in N$ is the the designated start symbol of G. The productions in R are pairs of the form $\alpha \rightarrow \beta$, where $\alpha \in N$ and $\beta \in (N \cup T)$. An empty right-hand side in a production is represented with symbol ϵ.

2.5 Semantic Attachments

In rule-to-rule approach [41], the production rules r from CFG G are extended with semantic attachments. To construct a semantic attachment, each production $r \in R$, $r : \alpha \rightarrow \beta_1 ... \beta_n$ is associated with a semantic rule $\alpha.sem$: $\{f(\beta_1.sem, ..., \beta_n.sem)\}$ to infer semantic ontological relationships. The semantic attachment $\alpha.sem$ states that the semantic representation assigned to production r contains a semantic function f that maps semantic attachments $\beta_i.sem$ to $\alpha.sem$, where each β_i, $1 \leqslant i \leqslant n$ is a constituent (terminal or non-terminal symbol) in production r. The semantic attachments for each production rule is shown in brackets $\{\ldots\}$ to the right of the production's syntactic constituents.

2.6 λ Calculus

λ calculus is a formal system in mathematical logic for expressing computation based on function abstraction and application using variable binding and substitution. λ calculus consists of constructing lambda terms and performing reduction operations on them. The following three rules give an inductive definition that can be applied to build all syntactically valid lambda terms:

- a variable, x, is itself a valid lambda term.
- if t is a lambda term, and x is a variable, then $(\lambda x.t)$ is a lambda term called lambda abstraction.
- if t and s are lambda terms, then (ts) is a lambda term called an application.

The beta reduction (β reduction rule states that an application of the form $(\lambda x.t)s$ reduces to the term $t[x = s]$.

2.6.1 Grounded Theory

A qualitative inquiry approach which involves applying specific types of codes to data through a series of coding cycles leading to development of a theory grounded in the data [59]. We use three main strategies [21] of this method throughout this work: (1) coding qualitative data; (2) memo-writing; and (3) theoretical sampling.

2.6.2 Word Embedding

Word embeddings are distributed representations of words as a vector in some m-dimensional space that helps learning algorithms achieve better performance by grouping similar words together [46], [5]. Each vector dimension represents some feature of the words' semantics in a corpus.

Currently, the two most popular word embedding models are Global Vectors (GloVe) [52] and Skip-gram [45]. GloVe trains word embedding vectors by constructing a word-to-word cooccurrence matrix. After filling the matrix with how frequently words co-occur, matrix factorization is used to determine the word embedding vector values for each word. For the Skip-gram model, a window size is defined before training begins. For each word in the corpus, the surrounding words (identified by the size of the window) are used as context for that word. This context is then used as input to a neural net-work that will modify the word's vector values. After visiting each word in the corpus, words should be grouped together in the vector space of the vocabulary based on their context. The closer together two words are in the vector space, the more semantically similar they are assumed to be. In this work, we adopt Word2Vec [3] , an implementation of the Skip-gram model to construct domain-specific word embeddings, which is discussed in Section 8.1.1.

2.6.3 Convolutional Neural Network

In general, convolutional neural networks (CNNs) are a kind of feed-forward network, specialized in processing data with a grid-like topology [42]. For CNNs, there are usually three major steps in a convolutional layer. The first step involves applying several convolutions to the input matrix to produce a set of linear activations [29]. This is done using a sliding window, called a kernel or filter that slides over the entire input matrix, thereby performing convolution for every set of elements. The second step applies a non-linear function to each linear activation produced by the previous step (e.g., tanh, relu, etc.). In the third step, different types of pooling functions (e.g., max pooling, average pooling, etc.) are applied to sets of areas, called neighborhoods, which cover the entire transformed input. Pooling is done to make the transformed input approximately invariant, which emphasizes the importance of the existence of a feature in the input over the specific location of that feature in the input [29]. The matrix result after these three steps is a representation of the main features of the input.

[3]https://code.google.com/archive/p/word2vec/

Chapter 3: RELATED WORK

3.1 Lexicons, Ontology, and Requirements Analysis

In requirements engineering, lexicons play an important role in reducing ambiguity and improving the quality of specifications [28]. Boyd et al. examined the design of constrained natural languages and their reliance on limited vocabulary to reduce ambiguity [9]. They proposed an automated technique to optimally constrain lexicons by introducing the concept of term replaceability. The value of their approach is that the lexicon becomes more easily evolvable over time. While a lexicon often consists of terminology and definitions, an ontology represents the semantic relationships between terms, including whether they are hypernyms or synonyms. Breitman and do Prado Leite describe how ontology can be used to analyze web application requirements [16]. Breaux et al. [12] utilize ontology consisting of actors and information types to infer data flow traces across privacy requirements of different vendors. In this work, ontology was needed to align terminology across different domains and vendor applications.

3.2 Ontology in Requirements Traceability to Code

Zimmeck et al. proposed an approach to identify the misalignments between data practices expressed in privacy policies and mobile app code without considering abstraction in policy statements [73]. Privacy policy annotations in this work yielded three bag-of-words for information types labeled as device ID, location, and contact information. For example, “IP address” is contained in the bag-of-words with label device ID. Therefore, the policy statement classifier in their approach labels policy statements using one of the three categories: device ID, location, and contact information. However, ignoring the actual information type in the statement can produce false negatives in misalignment detection tools. As an example, consider an application that mentions collecting “IP address” in the policy and calls an Android API method to retrieve “Android ID” in the app code. Since the Zimmeck et al.’s approach classifies “IP address” and “Android ID” as device IDs, this would be interpreted as an expected alignment, which in fact is a misalignment and the policy and code are inconsistent.

Salvin et al. [60] identify inconsistent app code with privacy policies by utilizing a manually constructed information type ontology. The ontology contains 365 unique information types from an analysis of 50 privacy policies. However, the cost of setting up the ontology is non-trivial as it requires manual assignment of semantic relations between information types and handcrafting the ontology. Identifying misalignments between natural language data practice descriptions and app code cannot be achieved without addressing ambiguity and abstraction of data type terminology. To address this challenge, we propose a method to construct a formal ontology that captures the semantic relationships between information types mentioned in these descriptions.

3.3 Lexical Ontologies

WordNet is a lexical database that contains English words grouped into nouns, verbs, adjectives, adverbs, and function words [24, 47]. Within each category, the words are organized by their semantic relations, including hypernymy, meronymy, and synonymy [24]. However, our analysis shows that only 14% of information types from a privacy policy ontology are found in WordNet, mainly because the lexicon is populated with multi-word, domain-specific phrases. Therefore, finding a category of information type along with its subordinate terms can be a challenging task for a requirement analyst. Our work aims to identify relationships between categories of information types with respect to hypernymy, meronymy, and synonymy in the privacy domain which can be reused in requirement analysis tasks.

3.4 Relationship Extraction and Classification Methods

Snow et al. [62] presented a machine learning approach using hypernym-hyponym pairs in WordNet to identify additional pairs in the parsed sentences of the Newswire corpus. This approach relies on the explicit expression of hypernymy pairs in text. Marti Hearst proposed six lexico-syntactic patterns to automatically identify hypernymy in text using noun phrases and regular expressions [34]. Evans et al. [23] 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. Chen et al. [19] presented an approach for gathering software terms and their morphological forms. Terms are limited to those in the software development domain. Roy et al. [58] presented an approach for inferring relationships between terms in math word problems.

In supervised paradigms, researchers have tried to classify the relationship between a pair of nominals in sentences by extracting features such as part-of-speech tags, shortest dependency path, and named entities [17, 31, 48, 70]. Performance among these methods depend on the quality of designed features [71]. To address this problem, deep learning models have been introduced that leverage a distributed representation of the words to reduce the number of handcrafted features. For example, Zeng et al. [71] proposed a deep learning model that captures the semantics of a sentence containing a pair of nominals by combining word and distance features using a convolution module. The features were used to classify the relationship between nominal pairs into four categories. This model extracts features using convolutional neural networks and outperforms the supervised models that use part-of-speech, stemming, and other lexical features with classifiers, such as SVM and MaxEnt. Zhou et al. [72] proposed a model that utilizes word embeddings, Bidirectional Long Short Term Memory (BLSTM), and attention-based neural networks for relation classification. Attention-based neural networks use the results from BLSTM to generate a single vector representing a sentence semantics. The softmax classifier is used to classify the relationships using the sentence vectors, which outperforms the model presented by [71].

The feature-based and neural network models mentioned above are used to extract the relationships between the annotated nominals in a given sentence. These approaches are all sentence dependent and fail to consider the semantic relations between phrases that are not in the same sentence. Therefore, our proposed work aims to model the semantics of two information types extracted from a pool of privacy policies and identifies the semantic relations between the them.

Chapter 4: ACQUIRING PRIVACY POLICY LEXICON

There is no standard method to build an ontology [67], yet, a general approach includes identifying the ontology purpose and scope; identifying key concepts leading to a lexicon; identifying relations between lexicon concepts; and formalizing those relations. A lexicon consists of terminology in a domain, whereas ontologies organize terminology by semantic relations [39]. Lexicons can be constructed using content analysis of source text, which yields an annotated corpus. This chapter describes our approach to build privacy policy lexicon. We use this approach to construct different lexicons including platform information lexicon and user-provided information lexicon. Section 4.1 describes the general approach to build a lexicon. Platform information lexicon and user-provided information lexicon are represented in Sections 4.2 and 4.3 respectively.

4.1 Lexicon Construction Approach

The mobile privacy policy lexicon (artifact A in Figure 4.1) was constructed using a combination of crowdsourcing, content analysis, and natural language processing (NLP). The lexicon construction method (see Figure 4.1) consists of 4 steps: (1) collecting privacy policies; (2) itemizing paragraphs in the collected privacy policies; (3) annotating the itemized paragraphs by crowd workers based on a specific coding frame; (4) employing an entity extractor developed by Bhatia and Breaux [6] to analyze the annotations and extract information types which results in an information type lexicon (artifact A in Figure 4.1). Steps 1-3 are part of a crowdsourced content analysis task based on Breaux and Schaub [13].

We use this approach to construct different lexicons using various privacy policies for apps in different domains. We use two different coding frames for annotating the privacy policies that captures platform information and user-provided information. More information about these lexicons is provided in the following sections.

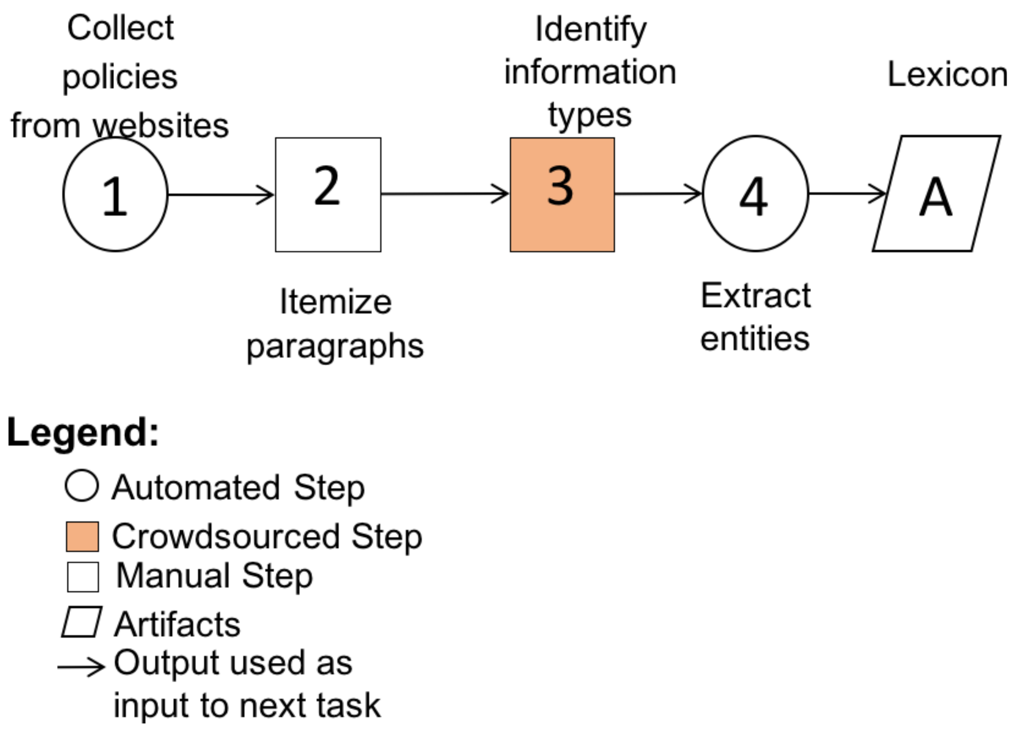
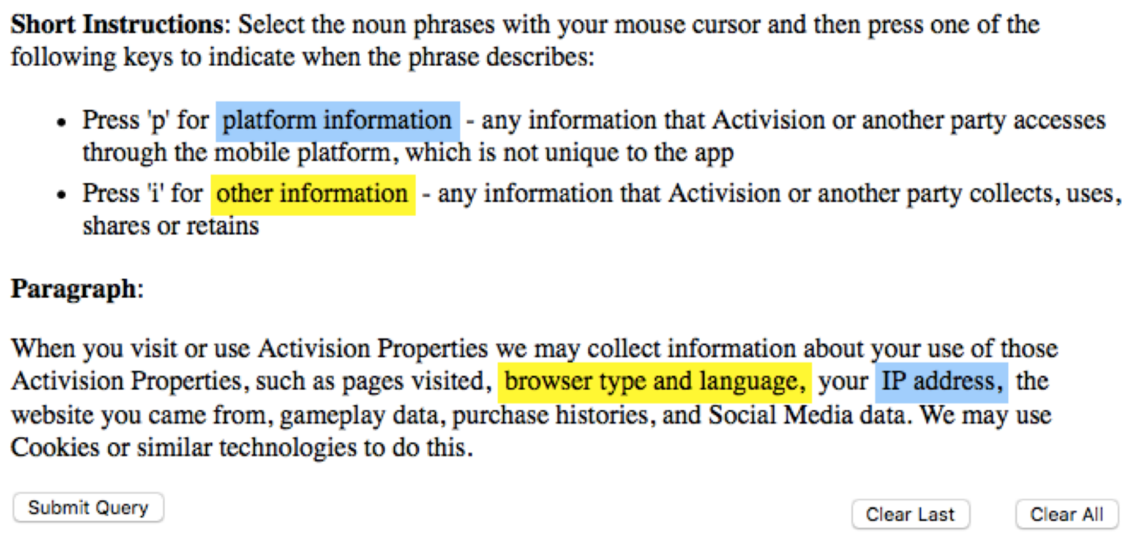


Figure 4.1: Overview of Lexicon Construction Method

4.2 Platform Information Lexicon

Using the approach discussed in Section 4.1, we constructed the platform information lexicon that is used by Slavin et al. [60] to identify the inconsistencies between privacy policies and app code based on the API method calls in the code.

In step 1 (see Figure 4.1), we selected the top 20 mobile apps across each of 69 sub-categories in Google Play. From this set, we selected apps with privacy policies, removing duplicate policies when different apps shared the same policy. Next, we selected only policies that match the following criteria: format (plain text), language (English), and explicit statements for privacy policy; yielding 501 policies, from which we randomly selected 50 policies. In step 2, the 50 policies were segmented into 120 word paragraphs using the method described by Breaux and Schaub [13]; yielding 5,932 crowd worker annotator tasks with an average 98 words per task for input to step 3. In step 3, the annotators select phrases corresponding to one of two category codes in a segmented paragraph as described below for each annotator task, called a Human Intelligence Task (HIT). An example HIT is shown in Figure 4.2.


Short Instructions: Select the noun phrases with your mouse cursor and then press one of the following keys to indicate when the phrase describes:

- Press 'p' for platform information - any information that Activision or another party accesses through the mobile platform, which is not unique to the app
- Press 'i' for other information - any information that Activision or another party collects, uses, shares or retains

Paragraph:

When you visit or use Activision Properties we may collect information about your use of those Activision Properties, such as pages visited, browser type and language, your IP address, the website you came from, gameplay data, purchase histories, and Social Media data. We may use Cookies or similar technologies to do this.

Figure 4.2: Example of Crowd Sourced Policy Annotation Task for Platform Information Types

- *Platform Information*: any information that the app or another party accesses through the mobile platform which is not unique to the app.

- *Other Information*: any other information the app or another party collects, uses, shares or retains.

These two category codes were chosen, because our initial focus is on information types that are automatically collected by mobile apps and mobile platforms, such as ”IP address," and ”location information." The other information code is used to ensure that annotators remain vigilant. In step 4, we selected only platform information types when two or more annotators agreed on the annotation to construct the lexicon. This number follows the empirical analysis of Breaux and Schaub [13], which shows high precision and recall for two or more annotators on the same HIT. Next, we applied an entity extractor [6] to the selected annotations to itemize the platform information types into unique entities included in the privacy policy lexicon.

Six privacy experts, performed the annotations. The cumulative time to annotate all HITs was 59.8 hours across all six annotators, yielding a total 720 annotations in which two or more annotators agreed on the annotation. The entity extractor reduced these annotations down to 356 unique information type names, which comprise the initial lexicon.

In chapter 5, we discuss how we utilized this lexicon to construct platform information ontology.

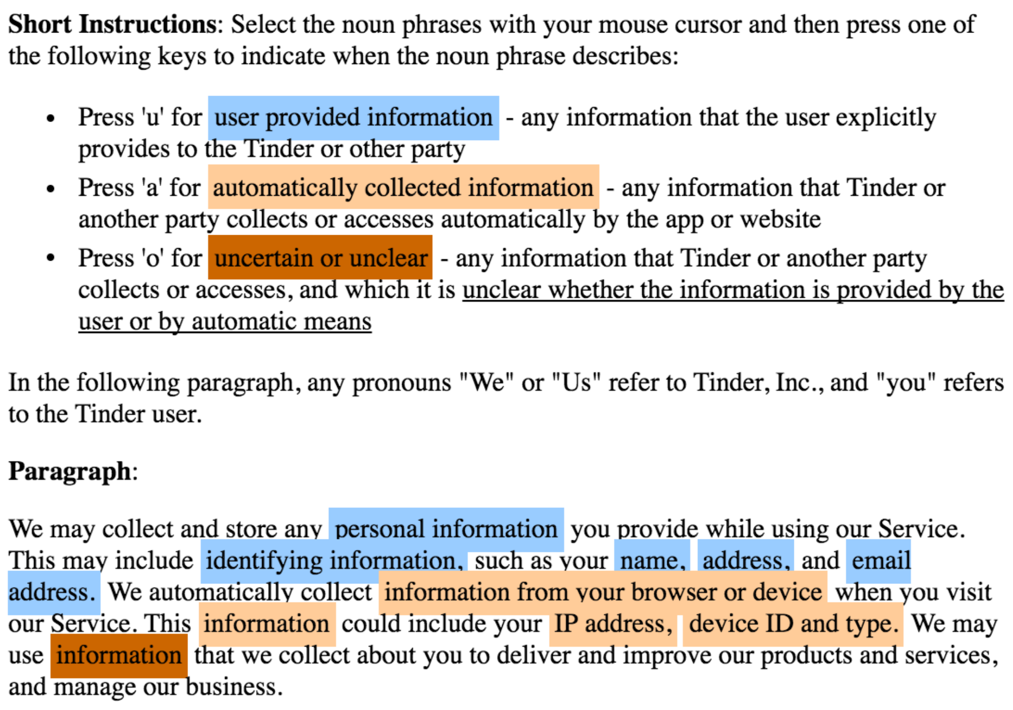
4.3 User-provided Information Lexicon

Using the approach discussed in Section 4.1, we constructed three different user-provided information lexicon for finance, health, and dating app domains that is used to identify the inconsistencies between privacy policies and user-provided information types collected through user interface input fields in apps.

To construct the lexicons, we first select five top apps in each of six sub-categories (personal budget, banks, personal health, insurance-pharmacy, casual and serious dating) of finance, health, and dating in Google Play, to yield 30 total apps for all three main categories. Next, we segment the privacy policies into 120 word paragraphs using the method described by [14], which yields annotation tasks from each policy. Figure 4.3 shows an example annotation task, wherein annotators are asked to annotate phrases based on the following coding frame: User-provided Information; Automatically Collected Information; and Uncertain or Unclear.

The user-provided information annotations describe types that are explicitly stated in the policies. However, policies do not always mention how or from whom they collect the information. For example, in Figure 4.3, it is unclear how "information" is collected. To build the privacy policy lexicon, we consider both annotations coded as user-provided information, and uncertain or unclear, in case the policy author described the user-provided collection in an unclear manner. We included the code for automatically collected information to ensure that annotators pay close attention about how information collection is described in the policy, since it is disjoint from user-provided information.

We collect annotations by recruiting five crowd workers from Amazon Mechanical Turk (AMT) to annotate each 120-word paragraph of the combined 30 privacy policies. Because this annotation task differed from [14], we also collected annotations for the same tasks from six privacy experts to evaluate the crowd worker lexicon. Among all annotations collected, we only add annotations


Short Instructions: Select the noun phrases with your mouse cursor and then press one of the following keys to indicate when the noun phrase describes:

- Press 'u' for user provided information - any information that the user explicitly provides to the Tinder or other party
- Press 'a' for automatically collected information - any information that Tinder or another party collects or accesses automatically by the app or website
- Press 'o' for uncertain or unclear - any information that Tinder or another party collects or accesses, and which it is unclear whether the information is provided by the user or by automatic means

In the following paragraph, any pronouns "We" or "Us" refer to Tinder, Inc., and "you" refers to the Tinder user.

Paragraph:

We may collect and store any personal information you provide while using our Service. This may include identifying information, such as your name, address, and email address. We automatically collect information from your browser or device when you visit our Service. This information could include your IP address, device ID and type. We may use information that we collect about you to deliver and improve our products and services, and manage our business.

Figure 4.3: Example of Crowd Sourced Policy Annotation Task for User-provided Information Types

to the lexicon where two or more annotators agreed on the annotation. This decision follows the study which shows high precision and recall for two or more annotators [14]. In the next step, we applied an entity extractor [6] to the selected annotations to itemize the information types into unique entities. Finally, the unique information types are added to the finance, health, or dating lexicon depending on which sub-category they belong to.

Table 4.1 shows the total HITs to collect information type annotations, average word count per HIT, total annotations collected from crowd workers and privacy experts, total unique information types extracted, and combined annotation time for crowd workers and privacy experts.

Overall, the average time to extract an information type from a privacy policy in health, finance, and dating is 10.6 minutes, 7.0 minutes, and 8.4 minutes, respectively. This time includes the additional time from privacy expert annotations needed to evaluate the method.

The lexicon quality is measured by the consensus between privacy expert and crowd worker annotations as measured by extracted, unique information types. In health, the privacy experts and crowd workers agreed on 105/198 unique information types. In addition, the privacy experts

Table 4.1: User-provided Information Lexicon analysis

	Health	Finance	Dating	Overall
Total HITs	141	52	141	334
Average Words per HIT	105	102	116	108
Total Annotations - Crowd Workers	739	309	868	1,916
Total Annotations - Privacy Experts	456	198	508	1,162
Total Unique Information Types	197	112	262	490
Annotation Time	34.7	13.1	36	84

missed 55 information types that the crowd workers annotated, and the crowd workers missed 34 information types that the privacy experts annotated. In finance, all annotators agreed on 69/112 information types, crowd workers annotated an additional 20 types, whereas privacy experts annotated an additional 23 types. In dating domain, all annotators agreed on 135/262 information types, crowd workers annotated additional 76 information types and privacy experts annotated 51 additional unique information types. Overall, the crowd workers generally identified 18-29% more information types, and privacy experts generally identified 17-20% more types. The consensus was 52-62% of types extracted.

In addition to comparing annotator performance, we compared the lexicon coverage across each domain. The health and finance lexicons share 32/278 phrases, health and dating share 45/415 phrases, and finance and dating share 27/347 phrases. This is an overlap of only 8-12% across three domains, which is due to the differences in policy focus and application features.

Section 5.3 describes our effort to construct three different ontologies using finance, health, and dating user-provided information lexicons.

Chapter 5: MANUAL ONTOLOGY CONSTRUCTION

We now describe our bootstrap method for constructing a formal ontology from an information type lexicon. This includes our choice of formalism, the tools used to express the ontology, and the construction method.

5.1 Manual Ontology Construction Methodology

The bootstrap begins with an initial ontology, wherein each lexicon phrase is subsumed by the $\top$ concept and no other relationship exists between phrases from a given lexicon. Next, each analyst follows these four steps: (1) they create two copies of the initial ontology $KB1$ and $KB2$, one for each analyst; (2) each analyst define subsumption and equivalence axioms for concept pairs using an ontology editor by making paired comparisons among the concepts in the ontology based on the heuristics defined bellow; (3) the two analysts compare their axioms in $KB1$ and $KB2$ to identify missing axioms and to compute the degree of agreement. Agreement is measured using the chance-corrected inter-rater reliability statistic Fleiss's Kappa; and (4) finally, two analysts meet to investigate the disagreements and reconcile the axioms in $KB1$ and $KB2$. The analysts re-calculate agreement after each reconciliation to measure the improvement due to reconciliation. Identifying semantic relationships is a heuristic-based procedure, wherein each analyst develops their own heuristics or rules for identifying relationships. The reconciliation step requires analysts to explicate and justify their choices, as well as to learn to accept or reject heuristics proposed by the other analyst. This method is subject to cognitive bias, including the proximity of concepts to each other in the alphabetical list, and to the recency with which the analysts encountered concepts for comparison [54].

The bootstrap method was piloted on five privacy policies and resulted in a set of seven heuristics that form a grounded theory. The heuristics explain why two concepts share an axiom in the ontology. For a pair of concepts C_1, C_2, the analysts assign an axiom with respect to a TBox T and one heuristic as follows.

- Hypernym (H): $C_1 \sqsubseteq C_2$, when concept C_2 is a general category of C_1, e.g., "device" is subsumed by "technology."

- Meronym (M): $C_1 \sqsubseteq C_2$, when concept C_2 is a part of C_1, e.g., "Internet protocol address" is subsumed by "Internet protocol."

- Modifier (D): $C_1\_C_2 \sqsubseteq C_2$ and $C_1\_C_2 \sqsubseteq C_1\_information$, when C_1 is modifying C_2, e.g., "unique device identifier" is subsumed by "unique information" and "device identifier."

- Plural (P): $C_1 \equiv C_2$, when C_1 is a plural form of C_2, e.g., "MAC addresses" is the plural form of "MAC address."

- Synonym (S): $C_1 \equiv C_2$, when C_1 is a synonym of C_2, e.g., "geo-location" is equivalent to "geographic location."

- Technology (T): $C_1 \equiv C_1\_information$, when C_1 is a technology, e.g., "device" is equivalent to "device information."

- Event (E): $C_1 \equiv C_1\_information$, when C_1 is an event, e.g., "usage" is equivalent to "usage information."

The bootstrap method and the seven heuristics are used to construct platform information ontology and user-provided information ontologies from platform information lexicon and user-provided information lexicons.

5.2 Platform Information Ontology

To construct the platform information ontology, two analysts conducted a *4-step heuristic evaluation* of the seven heuristics using the lexicon produced by the 50 mobile app privacy policies (platform information lexicon in Section 4.2)as follows: (1) two analysts separately apply the bootstrap method on a copy of the lexicon; (2) for each ontology, an algorithm extracts each expressed and inferred axiom between two concepts using the HermiT[1] reasoner; (3) each relationship assigned to

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

Table 5.1: Example Table Comparing Concept Relationships in Platform Information Ontology

LHS Concept	RHS Concept	Heuristics	Analyst1	Analyst2
web pages	web sites	S/M	Equiv	Sub
ads clicked	usage information	H	Sub	Sub
computer	platform	H	Super	Super
log information	system activity	M	None	Super
device type	mobile device type	D	Super	None
tablet	tablet information	T	None	Equiv

a concept pair appears in one column per analyst (in Table 5.1, see Analyst1, where 'Super' means the LHS (left-hand side)concept is a superclass of the RHS (right-hand side) concept, 'Sub' means subclass of, 'Equiv' means equivalence, and 'None' means no relationship); and (4) each analyst then separately reviews their assigned axiom type, choose the heuristic to match the assignment, and decides whether to retain or change their axiom type.

In Table 5.1, the left-hand side (LHS) concept is compared to the right-hand side (RHS) concept by Analyst1 and Analyst2, whose axiom types appear in their respective column, e.g., Analyst1 assigned "Equiv" to "web pages" and "web sites" and the heuristic "S" to indicate these two concepts are synonyms, whereas Analyst2 assigned "Sub" and heuristic "M" to indicate "web pages" is a part of "web sites."

Before and after step 3, we compute the Fleiss' Kappa statistic, which is a chance-corrected, inter-rater reliability statistic [25]. Increases in this statistic indicate improvement in agreement above chance. The result of evaluation of this ontology is presented in Section 5.4.

5.3 User-provided Information Ontology

Following the manual ontology approach, two analysts conducted a *4-step heuristic evaluation* of the seven heuristics on each of the user-provided information lexicons in finance, health, and dating domain (see Section 4.3)as follows: (1) two analysts separately apply the bootstrap method on a copy of each lexicon; (2) for each ontology, an algorithm extracts each expressed and inferred axiom between two concepts using the HermiT[2] reasoner; (3) each relationship assigned to

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

Table 5.2: Example Table Comparing Concept Relationships in Finance Domain

LHS Concept	RHS Concept	Heuristics	Analyst1	Analyst2
id	identification information	T/H	Equiv	Sub
account	payment account	D	Super	Super
account	financial information	H	Sub	Sub
account	account balance	M	Super	Super

a concept pair appears in one column per analyst (see Table 5.2, where 'Super' means the LHS (left-hand side) concept is a superclass of the RHS (right-hand side) concept, 'Sub' means subclass of, 'Equiv' means equivalence, and 'None' means no relationship); and (4) each analyst then separately reviews their assigned axiom type, choose the heuristic to match the assignment, and decides whether to retain or change their axiom type.

Table 5.2 shows results of step 3 during constructing an ontology for finance domain. In this table, the left-hand side (LHS) concept is compared to the right-hand side (RHS) concept by Analyst1 and Analyst2, whose axiom types appear in their respective column, e.g., Analyst1 assigned "Equiv" to "id" and "identification information" and the heuristic "T" to indicate these two concepts are equivalent, whereas Analyst2 assigned "Sub" and heuristic "H" to indicate "web pages" is a part of "web sites."

Before and after step 3, we compute the Fleiss' Kappa statistic, which is a chance-corrected, inter-rater reliability statistic [25]. Increases in this statistic indicate improvement in agreement above chance. Section 5.5 provides evaluation results for the three user-provided information ontologies.

5.4 Platform Information Ontology Evaluation and Results

The ontology was constructed using the bootstrap method and evaluated in two iterations (see Table 5.3): Round 1 covered 25/50 policies to yield 235 concept names and 573 axioms from the 4-step heuristic evaluation, and Round 2 began with the result of Round 1 and added the concepts from the remaining 25 policies to yield a total 368 concept names and 849 axioms. The resulting ontology produced 13 new concepts that were not found in the lexicon, because the analysts added tacit concepts to fit lexicon phrases into existing subsumption hierarchies. Table 5.3 presents the results of the number of "Super," "Sub," and "Equiv" axioms and "None" identified after the bootstrap method.

Table 5.3: Number of Ontological Relations Identified by Each Analyst during Each Round

Iteration	Analyst	Super	Sub	Equiv	None
Round 1	1	151	203	77	142
	2	157	172	78	166
Round 2	1	304	343	142	60
	2	313	352	151	33

Table 5.4 presents agreements, disagreements, the consensus (ratio of agreements over total axioms compared), and Kappa for the bootstrap method without reconciliation, called Initial, and after reconciliation, called Reconciled. The round 1 ontology began with 235 concepts and 573 relations from both analysts. The round 2 ontology extended the reconciled round 1 ontology with 132 new concepts.

Table 5.4: Number of Agreements, Disagreements, and Kappa for Concepts and Axioms per Round

	Round 1(concepts=235, axioms=573)		Round 2(concepts=368, axioms=849)	
	Initial	Reconciled	Initial	Reconciled
Agreed	252	543	743	808
Disagreed	321	30	106	12
Consensus	43.9%	94.8%	87.5%	98.4%
Kappa	0.233	0.979	0.813	0.977

Table 5.5 presents the number of heuristics by type that are assigned to relations by two analysts: (H)ypernym, (M)eronym, (A)ttribute, (P)lural, (S)ynonym, (T)echnology, and (E)vent. Multiple heuristics may apply to some phrases, such as comparing "device information" to "mobile device IP address," which can be compared using the H, A, and M heuristics depending on which order the analyst applies the heuristics.

Table 5.5: Number of Heuristics Applied by Type

Heuristic	Round 1		Round 2	
Analyst	1	2	1	2
Hypernym	349	354	248	357
Meronym	38	38	100	56
Modifier	29	36	108	39
Plural	13	13	12	13
Synonym	55	55	57	57
Technology	10	10	17	15
Event	0	0	4	3

5.5 User-provided Information Ontology Evaluation and Results

Separate ontologies were constructed for each domain (finance, health, and dating) where two analysts individually identify semantic relationships between phrases in a lexicon in one domain, followed by a reconciliation step to remove conflicts between annotators. To evaluate the quality of the ontology, we used Fleiss's Kappa to measure the degree of agreement above chance before and after each reconciliation step [14]. The average time per analyst to identify semantic relationships in health, finance and dating was 6 hours, 5 hours and 8 hours, respectively. The average time to reconcile disagreements were 3.7 hours, 2 hours and 5 hours, respectively.

In health, the resulting $KB1$ and $KB2$ for the two analysts contain 951 and 920 axioms, respectively. We obtained these results after two rounds of comparisons and reconciliations. The first comparison produced 491 axioms in disagreement and reconciliation reduced the disagreements to 78 axioms. The Fleiss Kappa after the first and second reconciliations were 0.77 and 0.80, respectively. In finance, the resulting $KB1$ and $KB2$ for the two analysts contain 590 and 582 axioms, respectively. The first comparison produced 292 axioms in disagreement and reconciliation reduced the disagreements to 43 axioms. The Fleiss Kappa after the first and second reconciliations were 0.83 and 0.92, respectively, showing a larger increase in agreement. In dating domain, the resulting $KB1$ and $KB2$ for the two analysts contain 1,049 and 1,289 axioms, respectively. The first comparison produced 569 axioms in disagreement and reconciliation reduced the disagreements to 146 axioms. The initial Fleiss Kappa before reconciliation was 0.17 which was increased to 0.79

after the first round of reconciliation.

To evaluate the ontology construction method, two different analysts, who we call examiners, independently applied the construction method to the finance lexicon. Before reconciliation, the Fleiss Kappa was 0.14, which is extremely low chance-agreement. After reconciliation, Kappa was increased to 0.83. The Kappa comparing the analyst- and examiner-constructed ontologies was 0.54. On inspection, the disagreement is comprised of 148/474 axioms. Among the 148 axioms, 74 axioms can be inferred using a syntactic analyzer, which automates the process of ontology construction by inferring semantics from lexicon phrases based on their syntax, alone (e.g., plural-singular forms that are equivalent for our purposes). Resolving the 74 axioms with a syntactic analyzer yields a Kappa of 0.75 between the analysts and examiners. Among the unresolved differences, the examiners found hypernymy relationships missed by the analysts, such as “deposited checks," which are a kind of “transaction.” We believe these differences are due to (1) the various interpretations of phrases by analysts and examiners; (2) the fatigue of comparing phrases; (3) and recency effects that both analysts and examiners experienced during ontology construction [54].

5.6 Application of Manually Constructed Ontology

In this section, I represent two studies that utilized the manually constructed ontologies from Sections 5.2 and 5.3. The studies benefit from ontologies in identifying the inconsistencies between privacy policy requirements and information collected through app code.

5.6.1 Application of Platform Information Ontology

Slavin et al. [60] utilize the ontology constructed using the manual approach from 50 privacy policies to detect inconsistencies between privacy policies and Android API method calls in app code.

They define two different kinds of inconsistencies between privacy policies and app code: (1) potential strong inconsistency occurs when the policy does not describe an app's data collection practice, and (2) potential weak inconsistency that occurs when the policy describes the data practice using abstract and vague terminology.

The ontology is used to formally reason about the meaning of terminology found in the API documents and privacy policies. For an API lexicon $\hat{A}$ and a privacy policy lexicon $\hat{P}$ consisting of unique terms (or concepts), the ontology is a Description Logic (DL) knowledge base KB that consists of axioms $C \sqsubseteq D$, which means concept C is subsumed by concept D, or $C \equiv D$, which means concept C is equivalent to concept D, for some concepts $C, D \in (A \cup P)$. Slavin et al. [60] use an API lexicon to map a method name m from an API document to a concept $A \in \hat{A}$. Therefore, they can infer (in a forward direction) all policy concepts $\{P | P \in \hat{P} \wedge KB \models P \sqsubseteq A \vee KB \models P \equiv A\}$. In this respect, they extract method names from method calls in a mobile app, then infer corresponding policy terms (among which at least one) that should appear in the mobile app's privacy policy. Similarly, they also reason in the backward direction to check which policy terms mentioned in the app's policy map to which method names corresponding to method calls in the app.

For example, in Figure 5.1, "IP Address" is a decedent of "Network Information", indicating that IP Address is a type of network information. The hierarchical nature of an ontology allows for transitive relationships that can be used for mapping API methods to phrases indirectly based on relationships between the phrases themselves.

They apply their inconsistency detection technique with the manually constructed ontology on 477 pairs of apps and privacy policies. The results show 402 potential inconsistencies in total, including 58 potential strong inconsistencies and 344 potential weak inconsistencies. The manual inspection revealed that, among these detected inconsistencies, 341 were true inconsistencies, including 74 potential strong inconsistencies and 267 potential weak inconsistencies. The detection of 267 true potential weak inconsistencies shows that the manually constructed information type ontology is helpful on differentiating potential weak inconsistencies from potential strong inconsistencies.

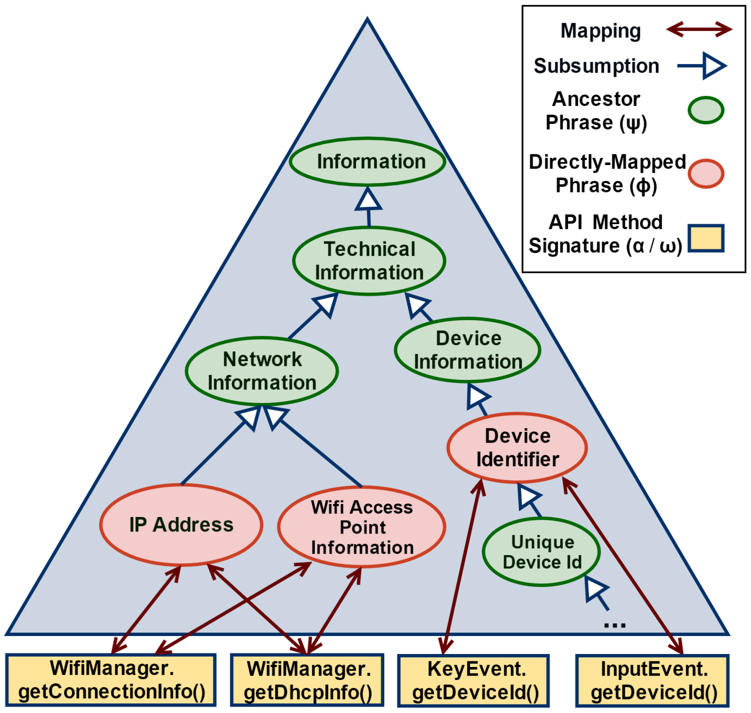


Figure 5.1: Abbreviated Ontology Example with Mapped API Methods

5.6.2 Application of User-provided Information Ontology

In another research we tried to trace privacy policy statements about collection of personal information to the information provided by users through app's graphical user interfaces. This work utilizes the user-provided information ontologies described in Section 5.3 to verify the compliance of app code to privacy policy requirements through: (1) identifying the abstraction in privacy policies, e.g., the privacy policy refers to collection of "personal information, " when the app collects users' "weight" or "age;" (2) identifying the information types that are collected through user input fields, but never mentioned in the privacy policies descriptions.

This work adopt the definition of *inconsistency*, which consists of: *potential weak inconsistency*, which occur when a policy refers to a vague or abstract information type that semantically includes a more specific type that was omitted from the policy; and *potential strong inconsistency*, which occur when the type is completely omitted from the policy. For example, if an app shares a user's medicine intakes, it would be considered a weak violation if the policy only states, "we collect *medical* information...." If the policy neglects to mention medical information as a collected type, then this omission is classified as a strong violation.

When detecting potential strong and weak inconsistencies, for an input field V whose collected information is sent to network, we first check whether V can be mapped to a concept word C in the ontology. If so, V is considered an input field collecting sensitive information. They further check whether C and C's super classes or equivalent concepts in the ontology appear in the privacy policy. If neither C nor C's super classes or equivalent concepts appear, a potential strong inconsistency is detected, and if any of C's super classes or equivalent concepts appears but C does not appear, a potential weak inconsistency is detected.

They evaluated their approach using 150 apps collected from the Google Play with privacy policies in three categories: finance, health, and dating. The highest ranked 50 apps that have privacy policies were selected for each category from Google Play using the category name as the search word. From this set, 30 privacy policies were used to construct the user-provided information ontology discussed in Section 5.3. The remaining 120 apps are utilized for the evaluation of the method. The approach detects 21 potential strong inconsistencies and 18 potential weak inconsistencies in 120 apps from three main domains.

5.7 Discussion and Conclusion

Our initial attempt to build an information type ontology requires comparing each information type phrase with every other phrase in the privacy policy lexicon, and assigning a semantic relationship to each pair. However, considering that a lexicon built from 50 policies contains more than 350 phrases, an analyst must make about 61,400 comparisons, which is over 200 hours of continuous comparison. To address this problem, Section 6 discusses a semi-automated semantic analysis method that uses lexical variation of information type phrases to infer ontological relations, such as hypernyms. Instead of performing paired comparisons, the analyst spends less than one hour typing the phrases, and then a set of semantic rules are automatically applied to yield a subset of all possible relations.

In the manual ontology construction approach, meronymy relationship is defined using subsumption ($\sqsubseteq$) in heuristic “M.” Using only subsumption for defining meronymy relations can imply transitivity between multiple consecutive meronymy relationships. However, meronymy relations are not necessary transitive considering different types of part-whole relationships based on Gretsl and Prinnebow [27,55]. To overcome this problem, we extend the relations' definitions in description logic by adding $partOf$ relationship that maps parts to wholes as follows: a part concept C that has a whole concept D, such that $T \models C \sqsubseteq (partOfD)$. The semi-automatic ontology construction approach which is presented in Section 6 utilizes this definition for meronymy relations.

Chapter 6: SYNTAX DRIVEN SEMANTIC ANALYSIS OF INFORMATION TYPES

The manual ontology construction approach discussed in Chapter 5 requires paired comparison of $n \times (n-1)/2$ for n phrases in a lexicon. To overcome this problem, in this chapter, we describe a semi-automated method that uses lexical variation of information type phrases to infer ontological relations, such as hypernyms. Instead of performing paired comparisons, the analyst spends less time assigning role types to the phrases, and then a set of semantic rules are automatically applied to yield a subset of all possible relations. These rules were first discovered in a grounded analysis of 356 information types during manually constructing the ontology [38], which is discussed in Section 5. To improve the semantic relations inferred using these initial set of rules, we established a ground truth by asking human subjects to perform the more time-consuming task of comparing phrases in the lexicon. We then compared the results of the semantic rules against these human interpretations, which led to identifying additional semantic rules. Finally, we evaluated the improved semantic rules using 109 unique information types extracted from six privacy policies, and human subject surveys to measure the correctness of the results produced by the semantic rules.

6.1 Syntax Driven Ontology Construction Method

The ontology construction method shown in Figure 6.1 consists of 3 steps: (1) pre-processing the phrases in the lexicon (artifact A in Figure 6.1); (2) assigning role types to each pre-processed phrase that yields information type phrases with associated role sequences; (3) automatically matching the type sequence of each phrase to a set of semantic rules to yield a set of ontology fragments consisting of hypernym, meronym, and synonym relationships.

6.1.1 Pre-processing Information Types

To construct an ontology using this semi-automated method, we utilize the platform information lexicon which includes platform information types from 50 privacy policies (see Section 4.2). In

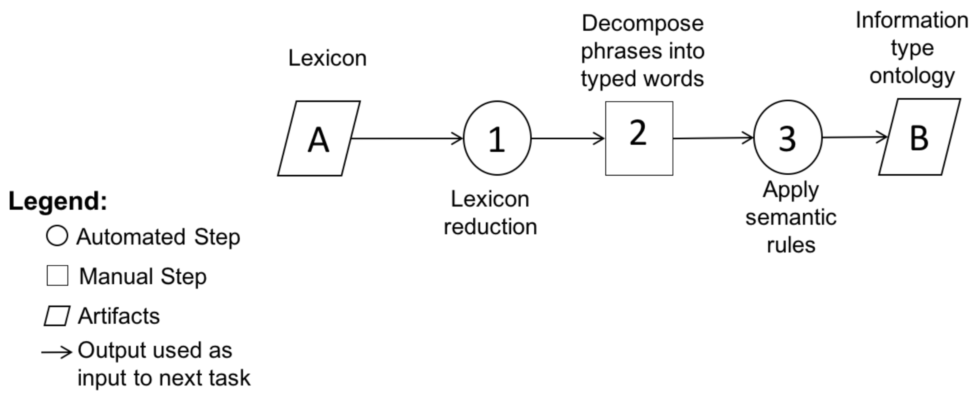

Lexicon
Decompose phrases into typed words
Information type ontology
A
1
2
3
B
Lexicon reduction
Apply semantic rules
Legend:
Automated Step
Manual Step
Artifacts
Output used as input to next task

Figure 6.1: Overview of Syntax Driven Ontology Construction Method

step 1, the lexicon is reduced using following approach:

- Plural nouns were changed to singular nouns, e.g., “peripherals” is reduced to “peripheral.”
- Possessives were removed, e.g., “device’s information” is reduced to “device information.”
- Suffixes “-related,” “-based,” and “-specific” are removed, e.g., “device-related” is reduced to “device.”

This approach reduced the lexicon to 335 information types.

6.1.2 Semantic Role Typing of Lexicon Information Types

Figure 6.2 shows an example phrase, “mobile device IP address” that is decomposed into the atomic phrases: “mobile,” “device,” “IP,” “address,” based on a 1-level, shallow typology. The typology links atomic words from a phrase to one of five roles: (M) modifiers, which describe the quality of a thing, such as “mobile” and “personal;” (T) things, which is a concept that has logical boundaries and which can be composed of other things; (E) events, which describe action performances, such as “usage,” “viewing,” and “clicks;” (G) agents, which describe actors who perform actions or possess things; (P) property, which describes the functional feature of an agent, place or thing, such as “date,” “name,” “height;” and (α) which is an abstract type that indicates “information,” “data,” “details,” and any other synonym of “information.” In an information type ontology, the concept that corresponds to the α type is the most general, inclusive concept.

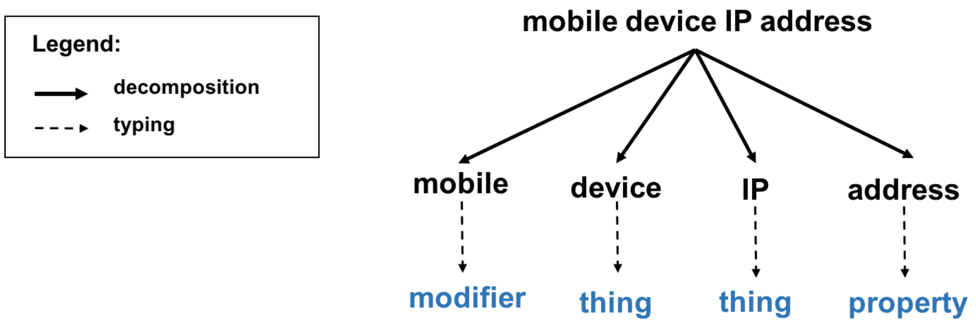


Figure 6.2: Example Lexicon Phrase, Grouped and Typed

In step 2, the analyst reviews each information type phrase in the lexicon and assigns role types to each word. The phrase typing is expressed as a continuous series of letters that correspond to the role typology. Unlike quadratic number of paired comparisons required to identify relationships among lexicon phrases, this typing step is linear in the size of the lexicon. Furthermore, word role types can be reused across phrases that reuse words to further reduce the time needed to perform this step.

Next, the semantic rules that are applied to the typed information types in the lexicon are introduced.

6.1.3 Automated Lexeme Variant Inference

We now describe step 3, which takes as input the typed, atomic phrases produced in step 2 to apply a set of semantic rules to infer variants and their ontological relationships, which we call variant relationships. Rules consist of a type pattern and an inferred ontological relationship. The type pattern is expressed using the typology codes described in Section 6.1.2. The rules below were discovered by classifying the platform information lexicon phrases using the typology as a second-cycle coding, which is a qualitative research method [59]. Subscripts indicate the order of same-typed phrases in asymmetric ontological relations:

Hypernymy Rules

H1. M_α implies that $MM\_\alpha \sqsubseteq \alpha$, e.g., "unique information" is a kind of "information."

H2. $M_1\_M_2\_\alpha$ implies that $M_1\_M_2\_\alpha \sqsubseteq (M_1\_alpha \sqcup M_2\_alpha)$, e.g., "anonymous demographic information" is a kind of "anonymous information" and "demographic information."

H3. $M\_T_1\_T_2$ implies $M\_T_1\_T_2 \sqsubseteq (M\_\alpha \sqcup T_1\_T_2)$ and $T_1\_T_2 \sqsubseteq partOf M\_T_1$, e.g., "mobile device hardware" is a kind of "mobile information," "device hardware," and "device hardware" is a part of "mobile device."

H4. M_T_α implies $M\_T\_\alpha \sqsubseteq (M\_\alpha \sqcup T\_\alpha)$, e.g., "mobile device information" is a kind of "mobile information" and "device information."

H5. M_T_P implies $M\_T\_P \sqsubseteq M\_\alpha$ and $M\_T\_P \sqsubseteq partOf M\_T$ and $T\_P \sqsubseteq partOf M\_T$, e.g., "mobile device name" is a kind of "mobile information" and a part of "mobile device" and "device name" is a part of "mobile device."

H6. M_G_α implies that $M\_G\_\alpha \sqsubseteq (M\_\alpha \sqcup G\_\alpha)$, e.g., "aggregated user data" is a kind of "aggregated data" and "user data."

H7. T_α implies $T\_\alpha \sqsubseteq \alpha$, e.g., "device information" is a kind of "information."

H8. $T_1\_T_2\_\alpha$ implies $T_1\_T_2\_\alpha \sqsubseteq (T_1\_\alpha \sqcup T_2\_\alpha)$, e.g., "device log information" is a kind of "device information" and "log information."

H9. G_α implies that $G\_\alpha \sqsubseteq \alpha$, e.g., "user information" is a kind of "information."

H10. G_T implies that $G\_T \sqsubseteq (G\_\alpha \sqcup T)$, e.g., "user content" is a kind of "user information" and "content."

H11. G_P implies that $G\_P \sqsubseteq (G\_\alpha \sqcup P)$, e.g., "user name" is a kind of "user information" and "name."

H12. E_α implies that $E\_\alpha \sqsubseteq \alpha$, e.g., "usage data" is a kind of "data."

H13. T_E implies that $T\_E \sqsubseteq (T \sqcup E \sqcup E\_lemma)$, e.g., "page viewed" is a kind of "page,""viewed," and "view."

Meronymy Rules

M1. $T_1\_T_2$ implies $T_1\_T_2 \sqsubseteq partOf T_1$ and $T_1\_T_2 \sqsubseteq T_2$, e.g., "Internet protocol address" is a part of "Internet protocol" and is a kind of "address."

M2. $T_1\_M\_T_2$ implies $T_1\_M\_T_2 \sqsubseteq partOf T_1$ and $M\_T_2 partOf T_1$, e.g., "device unique ID" is part of "device," and "unique ID" is a part of "device."

M3. T_P implies $T\_P \sqsubseteq T$ and $T\_P \sqsubseteq P$, e.g., "device name" is a part of "device" and a kind of "name."

M4. E_T implies that $E\_T \sqsubseteq partOfE$ and $E\_T \sqsubseteq T$, e.g., "advertising identifier" is a part of "advertising" and a kind of "identifier."

M5. E_P implies $E\_P \sqsubseteq partOfE$ and $E\_P \sqsubseteq P$, e.g., "click count" is part of "click" and a kind of "count."

M6. T_E_α implies that $T\_E\_\alpha \sqsubseteq partOfT$ and $T\_E\_\alpha \sqsubseteq (T\_\alpha \sqcup E\_\alpha)$, e.g., "language modeling data" is a part of "language" and a kind of "language data" and "modeling data."

M7. $M_1\_T_1\_M_2\_T_2$ implies $M_1\_T_1\_M_2\_T_2 \sqsubseteq partOfM_1\_T_1$ and $M_1\_T_1\_M_2\_T_2 \sqsubseteq M_2\_T_2$, e.g., "mobile device unique identifier" is a part of "mobile device" and a kind of "unique identifier."

M8. $T_1\_E\_T_2$ implies that $T_1\_E\_T_2 \sqsubseteq partOfT_1\_E$ and $T_1\_E\_T_2 \sqsubseteq (E\_T_2 \sqcup T_1\_information \sqcup T_2\_information)$, e.g., "Internet browsing behavior" is a part of "Internet browsing" and a kind of "browsing behavior" and "Internet information" and "behavior information."

M9. T_E_P implies that $T\_E\_P \sqsubseteq partOfT\_E$ and $T\_E\_P \sqsubseteq (E\_P \sqcup T\_\alpha \sqcup P)$, e.g., "website activity date" is a part of "website activity" and a kind of "activity date," "website information," and "date information."

Synonymy Rules

S1. T implies $T \equiv T\_\alpha$, e.g., "device" is a synonym of "device information."

S2. P implies $P \equiv P\_\alpha$, e.g., "name" is a synonym of "name information."

S3. E implies $E \equiv (E\_\alpha \sqcup E\_Lemma)$, e.g. "views" is a synonym of "views information" and "view."

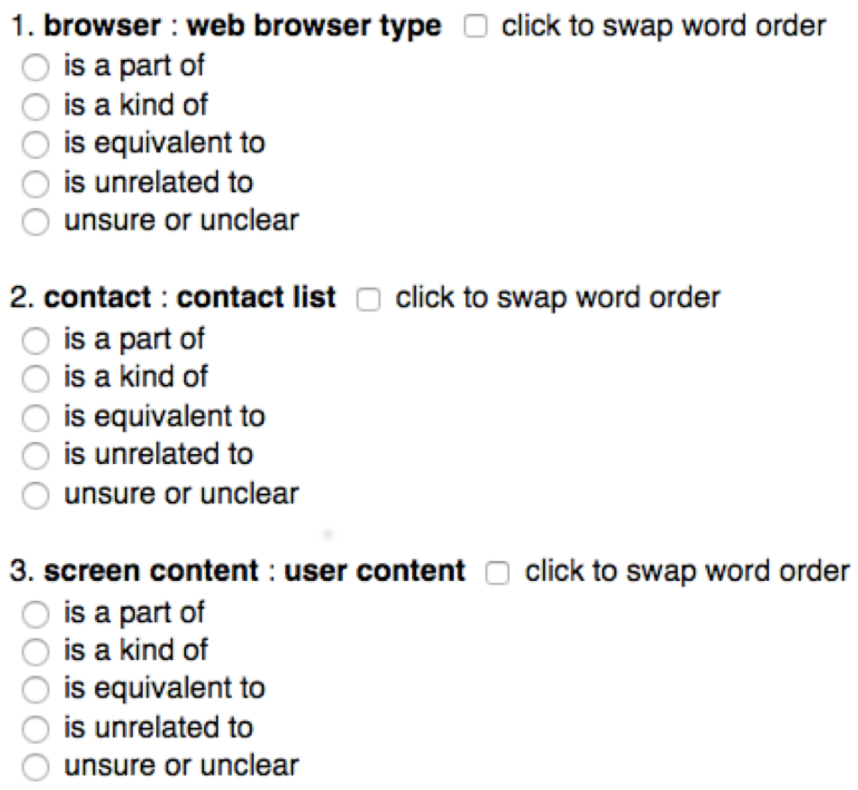
S4. G implies $G \equiv G\_\alpha$, e.g., "user" is a synonym of "user information."

The automated step 3 applies the rules to phrases and yields variant relationships for evaluation in two steps: (a) the semantic rules are matched to the typed phrases to infer new candidate phrases and relations; and (b) for each inferred phrase, we repeat step (a) with the inferred phrase. The technique terminates when no rules match a given input phrase. For example, in Figure 6.2, we perform step (a) by applying the rule H3 to infer that "mobile device IP address" is a kind of "device IP address" and "mobile information." However, the phrase "device IP address" is not in the lexicon, i.e., it is potentially an implied or tacit concept name. Thus, we re-apply the rules to "device IP address." Rule M3 matches this phrase's typing to infer that "device IP address" is part of "device IP" and a kind of "address." Since "device IP" is not in the lexicon, we re-apply the rules to this phrase. Rule M1 matches the role type sequence of this phrase that yields "device IP" is a part of "device" and "device IP" is a kind of "IP." Both "device" and "IP" are explicit concept names in the lexicon. Therefore, we accept both inferences for further evaluation. The axioms from applying and re-applying the rules to the explicit and tacit concepts names yield ontology fragments. We evaluate these axioms using the individual preference relationships described in the next section.

6.2 Experiment Setup

In psychology, preferences reflect an individual's attitude toward one or more objects, including a comparison among objects [61]. We designed a survey to evaluate and improve the ontological relationship prospects produced by step 3. We used 50 privacy policies and 335 unique information types in platform information lexicon as training set to improve the semantic rules. Because the prospects produced by the semantic rules all share at least one common word, we asked 30 human subjects to compare each $(335 - 334)/2 = 2,365$ phrase pairs from the lexicon that share at least one word. The survey asks subjects to classify each pair by choosing a relationship from among one of the following six options:

- s: Phrase A is subsumed by phrase B in pair (A, B)
- S: Phrase B is subsumed by phrase A in pair (A, B)
- P: Phrase A is part of Phrase B in pair (A, B)
- W: Phrase B is part of Phrase A in pair (A, B)
- E: Phrase A is equivalent to phrase B in pair (A, B)



1. **browser** : **web browser type** ☐ click to swap word order
   - ○ is a part of
   - ○ is a kind of
   - ○ is equivalent to
   - ○ is unrelated to
   - ○ unsure or unclear

2. **contact** : **contact list** ☐ click to swap word order
   - ○ is a part of
   - ○ is a kind of
   - ○ is equivalent to
   - ○ is unrelated to
   - ○ unsure or unclear

3. **screen content** : **user content** ☐ click to swap word order
   - ○ is a part of
   - ○ is a kind of
   - ○ is equivalent to
   - ○ is unrelated to
   - ○ unsure or unclear

Figure 6.3: Example Survey Questions to Collect Relation Preferences

- U: Phrase A is unrelated to phrase B in pair (A, B)

Figure 6.3 presents a survey excerpt: the participant checks one option to indicate the relationship, and they can check a box to swap the word order, e.g., in the first pair, the subject can check the box to indicate that "web browser type" is part of "browser." We recruited 30 participants to compare each phrase using Amazon Mechanical Turk, in which five pairs were shown in one Human Intelligence Task (HIT). Qualified participants completed over 5,000 HITs, had an approval rating of at least 97%, and were located in the United States.

The participant results are analyzed to construct a ground truth (GT) TBox in Description Logic. In the results, participants can classify the same phrase pair using different ontological relations. There are several reasons that explain multiple ontological relations for each pair: participants may misunderstand the phrases, or they may have different experiences that allow them to perceive different interpretations (e.g., "mac" can refer to both a MAC address for Ethernet-based routing, and a kind of computer sold by Apple, a manufacturer). To avoid excluding valid interpretations, we built a multi-viewpoint TBox that accepts multiple, competing interpretations. For the entire survey results, we define valid interpretations for a phrase pair to be those interpretations where the observed number of responses per category exceeds the expected number of responses in a Chi-square test, where $p<0.05$, which means there is at least a 95% chance that the elicited response counts are different than the expected counts. The expected response counts for an ontological relationship are based on how frequently participants chose that relationship across all comparisons. We constructed a multi-viewpoint TBox GT as follows: for each surveyed pair, we add an axiom to GT for the relation category, if the number of participant responses is greater than or equal to the expected Chi-square frequency; except, if the number of unrelated responses exceeds the expected Chi-square frequency, then we do not add any axioms. We published the ground truth dataset that includes phrase pairs, the ontological relation frequencies assigned by participants to each pair, and the Chi-square expected values for each relation per pair.

We measure the number of true positives (TP), false positives (FP), and false negatives (FN) by comparing the variant relationships with the ground truth ontology to compute Precision = TP/(TP+FP) and Recall = TP/(TP+FN). A variant relation is a TP, if it is logically entailed by GT, otherwise, that relationship is a FP. For all phrase pairs with valid interpretations that do not match an inferred variant relationship, we count these as FN. We use logical entailment to identify true positives, because subsumption is transitive and whether a concept is a hypernym to another concept may rely on the transitive closure of that concept's class relationships. The results from improving the semantic rules using the training dataset is presented in Section 6.3. The approach for building the test set to evaluate the final rule set is also presented in Section 6.3.

6.3 Evaluation and Results

We now describe the results from applying the semantic rules to the 335 unique information types as part of our training approach. The evaluation consists of an initial rule set based on analyzing the lexicon, and extensions to the initial rule set based on the individual preferences for ontological relations among variants.

Table 6.1: Evaluations of Relations Using Initial and Extended Rule Sets on TBox RGT

Rule Set	Precision	Recall
Initial rules	0.988	0.286
Extended rules	0.996	0.560

6.3.1 Preference Relations with Initial Rule Set

We began with a set of 17 rules that summarized our intuition on 335-phrase lexicon for variant relationship inference. After typing and decomposition, the technique yields 139 explicit concept names from the original lexicon, 197 potential tacit concept names, and 1,369 total axioms. Comparing the inferred relations with the individuals' preferences in the training ground truth TBox RGT, results in 0.988 precision and 0.286 recall. Overall, the method correctly identifies 359/1,252 hypernyms, meronyms, and synonyms in the TBox RGT. Next, we discuss our analysis of false positives and the TBox RGT to identify new rules that help explain how individuals perceive semantic relations.

6.3.2 Preference Relations with Extended Rule Set

The extended rule set consists of the initial rules and nine additional rules to improve the semi-automated method, in addition to a new meronymy-inferred relationship to rule H3 as defined in Section 6.1. The technique yields 194 explicit concept names, 289 potential tacit concept names, and 2,495 total axioms. The ontology fragments computed by applying the extended rule set can be found online in the OWL format[1]. Table 6.1 shows the precision (Prec.) and recall (Recall) for the semi-automated method with the initial and extended rule sets. Overall, the extended rule set correctly identifies 810/1,446 of hypernyms, meronyms, and synonyms in the TBox RGT. Also, the recall is improved to 0.560 with the extended rule set.

We observed that 493/636 of false negatives (FNs) depend on semantics beyond the scope of the 6-role typology. For example, the TBox RGT shows the participants agreed that "mobile phone" is a kind of "mobile device," possibly because they understood that "phone" is a kind of

[1]http://gaius.isri.cmu.edu/dataset/plat17/variants.owl

"device." We observed that 21/493 of semantically related FNs exclusively concern synonyms that require additional domain knowledge, e.g., "postal code" is equivalent to "zip code," or in the case of acronyms, "Internet protocol address" is equivalent to "IP address." Moreover, 8/493 of semantically related FNs exclusively concern meronymy, e.g., "game activity time" is a part of "game system." Also, only 1/493 of semantically related FNs is exclusively mentioned for hypernymy: "forwarding number" is a kind of "valid mobile number." Finally, 463/493 of semantically related FNs can have multiple valid interpretations (meronymy, hypernymy, and synonymy) in TBox RGT.

In addition, we discovered that 73/636 of FNs were due to individual preference-errors that were inconsistent with the automated method, e.g., individual preferences identified "mobile device identifier" equivalent to "mobile device unique identifier," which ignores the fact that an identifier is not necessarily unique. Finally, we identified 70/636 relations that can be identified by introducing new semantic rules.

The TBox RGT also contains a special relationship identified by individuals between 40 pairs that we call part-of-hypernymy. For example, individuals identified "device id" as a part of "mobile device," because they may have assumed that mobile device (as a hyponym of device) has an ID. Therefore, we extended rules H3 and H5 to infer part-of-hypernymy in the extended rule set.

6.3.3 Method Evaluation

To evaluate our extended rule set, we randomly selected six additional privacy policies from the pool of 501 policies discussed in Section 4.2. We used the similar approach and annotators from Section 4.2 to extract the unique information types and construct the test lexicon. The resulting 109 information types were then reduced, typed, and analyzed by the extended rule set, resulting in 76 explicit concept names, 139 potential tacit concept names, and 831 total axioms. We acquired the preference relations for the test lexicon by surveying 213 phrase pairs resulting in test ground truth TBox TGT using the method discussed in Section 6.3[2]. In further analysis, the relations in TBox TGT were compared with the relations provided by the extended rule set. Table 6.2presents

[2]http://gaius.isri.cmu.edu/dataset/plat17/study-utsa-prefs-test-set.csv

Table 6.2: Evaluations of Relations Using Extended Rule Set on TBox TGT

Rule Set	Precision	Recall
Extended rules based on preferences	1.00	0.593

the precision and recall for this analysis. The ontology fragments computed using the extended rule set are online in OWL[3]

We observed that 44/54 of false negatives (FNs) in the test set depend on semantics beyond the scope of the role typology. We published a list of these concept pairs, including the human preferences[4]. Some examples include: "device open UDID" as a kind of "device identifier," "in-app page view" as a kind of "web page visited," and "page viewed" as equivalent to "page visited." We also observed 7/54 of FNs that requires introducing six new rules.

6.4 Discussion and Future Work

We now discuss our results, including our interpretation of the extended rule set results.

When comparing the ontology fragments with the individual preferences, we observe that preferences imply axioms that are not identified by the semi-automated method and are therefore listed as FNs in both training and testing. The preferences are highly influenced by individual interpretations of relations between two phrases. Analyzing these FNs, we identified four cases where individuals report incorrect interpretations:

1. Modifiers' roles in a phrase are ignored and an equivalent relationship is identified for a pair of phrases, e.g., "unique ID" and "ID."

2. Different modifiers are identified as equivalent, e.g., "approximate location information" and "general location information."

3. The superordinate and subordinate phrases relationship is ignored and an equivalent relation is identified, e.g., "hardware" and "device," "iPhone" and "device."

[3]http://gaius.isri.cmu.edu/dataset/plat17/variants-test-set.owl
[4]http://gaius.isri.cmu.edu/dataset/plat17/supplements-test-set.csv

4. Information as a whole that contains information is confused with information as a subordinate concept in a super-ordinate category, e.g., “mobile application version” is a part of, and a kind of, “mobile device information.”

One explanation for the inconsistencies is that individuals conflate interpretations when comparing two phrases as a function of convenience. Without prompting individuals to search their memory for distinctions among category members (e.g., iPhone is different from Android, and both are kinds of device), they are inclined to ignore these distinctions when making sense of the comparison. In requirements engineering, this behavior corresponds to relaxing the interpretation of constraints or seeking a narrower interpretation than what the natural language statement implies. When relaxing constraints, stakeholders may overlook requirements: e.g., if “actual location” and “physical location” are perceived as equivalent, then stakeholders may overlook requirements that serve to more closely approximate the “actual” from noisy location data, or requirements to acquire location from environmental cues to more closely approximate a “physical” location. Furthermore, this behavior could yield incomplete requirements, if analysts overlook other, unstated category members.

In future work, we envision a number of extensions. To increase coverage, we propose to formalize the rules as a context-free grammar with semantic attachments using rule-to-rule hypothesis [4]. To improve typing, we explore identify role types associated with part-of-speech (POS) tagging and English suffixes. However, preliminary results on 335 pre-processed phrases from the training lexicon shows only 22% of role type sequences can be identified using POS and English suffixes. Therefore, instead relying on POS and suffix features [71], we envision using deep learning methods to learn the features for identifying the semantic relations between phrases.

Chapter 7: SEMANTIC INFERENCE USING RULE-TO-RULE HYPOTHESIS

Information types are frequently variants of a common lexeme, e.g, “mobile device” is a morphological variant of “device” through formation of a new lexeme. In this section, we propose an ontology construction method, which is a formalization of the rules discussed in Section 6.1.3. To this end, we first decompose each information type phrase into smaller units, called morphological variants. For example, “device IP address” is a morphological variant of “mobile device IP address.” Next, we use syntax analysis and a context-free grammar (CFG) over typed constituents to infer semantic relations between the information type and its variants. This proposed method is based on rule-to-rule hypothesis [4], which is augmenting CFG production rules with semantic attachments. For example, we can infer that “mobile device IP address” is a kind of “device IP address” by applying semantic attachments on “mobile device IP address” and its morphological variants.

We evaluate our ontology construction method as follows. First, we infer ontological relations from 356 information types in the platform lexicon, which we call L_1(see Section 4.2) using the context-free grammar and semantic attachments and compare the results with human classification results. The results are obtained using preference study, in which crowd workers select the best semantic relation between two information types based on their background knowledge and experience. Our analysis shows that our approach can infer ontological relations with 99% precision and 67% recall compared to human preferences.

To evaluate the coverage and generalization of the CFG and semantic attachments, we use a new lexicon called L_2 containing 1,853 general information types extracted from 30 policies [23]. We sampled 1,138 information types and 2,283 information type pairs from this lexicon and compared the inferred ontological relations from our approach with experimentally collected human preferences. The analysis shows our approach can infer relations with 99% precision and 91% recall.

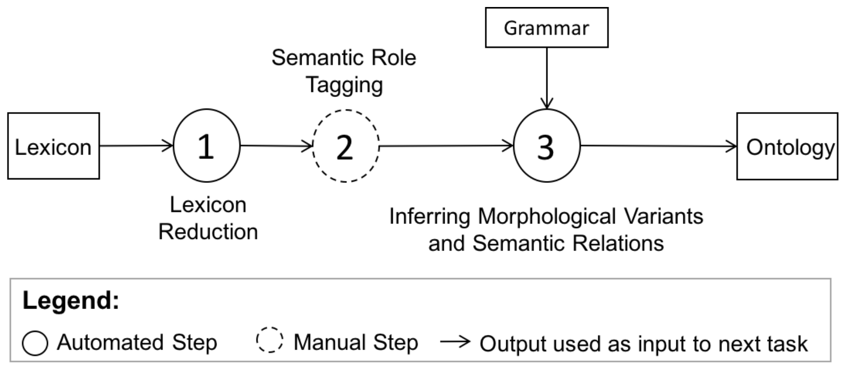


Figure 7.1: Ontology Construction Method Overview using Rule-to-Rule Approach

This chapter is organized as follows. In Section 7.1 we introduce our method. In Section 7.2, we present the evaluation and results, followed by concluding remarks in Section 7.3.

7.1 Ontology Construction Method using rule-to-rule approach

Figure 7.1 presents our method overview given a privacy policy lexicon. This figure is summarized as follows: in step 1, information types in a lexicon are pre-processed and reduced; in step 2, an analyst manually assigns semantic roles to the words in each reduced information type, a step that is linear in effort in the size of the lexicon; in step 3, a context-free grammar (CFG) and its semantic attachments are used to automatically infer morphological variants and ontological relations.

The production rules that comprise the CFG and that are introduced in this chapter are used to formalize and analyze the syntax of a given information type. Moreover, under a generative treatment of morphology [20], our CFG produces variants of a given phrase, which are necessary to build an ontology and overcome the abstraction problem in natural text. To infer ontological relationships, we implement the rule-to-rule hypothesis [4] by mapping each each production rule in the CFG to its semantic counterpart, presented using λ-calculus. We now discuss each step in our method.

7.1.1 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.”

7.1.2 Semantic Role Tags

Given the reduced lexicon as the input, step 2 consists of tagging each word in a phrase with one of five semantic roles: (m) modifiers, which describe the quality of a head word, such as “mobile” and “personal;” (t) thing, which is a concept that has logical boundaries and can be composed of other things; (e) events, which describe action performances, such as “usage,” “viewing,” and “clicks;” (a) agents, which describe actors who perform actions or possess things; (p) property, 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 the x tag that is the most general, inclusive concept [37].

One approach to tagging words could be to use part-of-speech (POS) tags, that are commonly used for natural language phrases and sentences [41]. 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 will frequently be nouns. However, our earlier analysis of lexicon L_1 shows that only 22% of tag sequences can be identified using POS and English suffixes [37]. Therefore, we rely on a manual tagging of words. As the only manual step in the approach, an analyst only needs to classify a number of words that is linear in the size of lexicon, which is a far smaller effort than the quadratic number of pairwise comparison required to manually construct an ontology.

The information type tagging is expressed as a continuous series of letters that correspond to

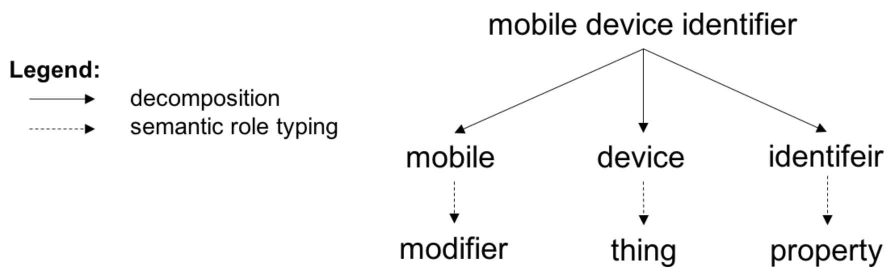


Figure 7.2: Example of Lexicon Phrase, Tokenized and Tagged

the semantic roles. Figure 7.2 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.

7.1.3 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.

Table 7.1: Context-Free Grammar for Syntax Analysis

<S>→<Modified1> | <Modified2> | <Final> | x
<Modified1>→ m<Modified1> | m<Modified2> | m<Final> | mx
<Modified2>→ a<Final> | e<Final> | a<Info>
<Final>→ t<Part> | t<Info> | e<Info> | p
<Part>→<Modified1> | <Modified2> | <Final>
<Info>→ x|ϵ

In Figure 7.1, step 3 begins by processing the tagged information type phrases from the reduced lexicon using the CFG shown in Table 7.1. The CFG represents the antecedent and subsequent tags used to infer morphological variants from a given information type in the lexicon. This grammar was discovered by applying grounded analysis to the tag sequences of all information types in lexicon L_1. Notably, the grammar distinguishes between four kinds of tag sub-sequences: (1) a type that is modified by a modifier, called Modified1; (2) a type that is modified by an agent (e.g., "user" or "company") or event (e.g., "click" or "crash"), called Modified2; (3) a Final type that

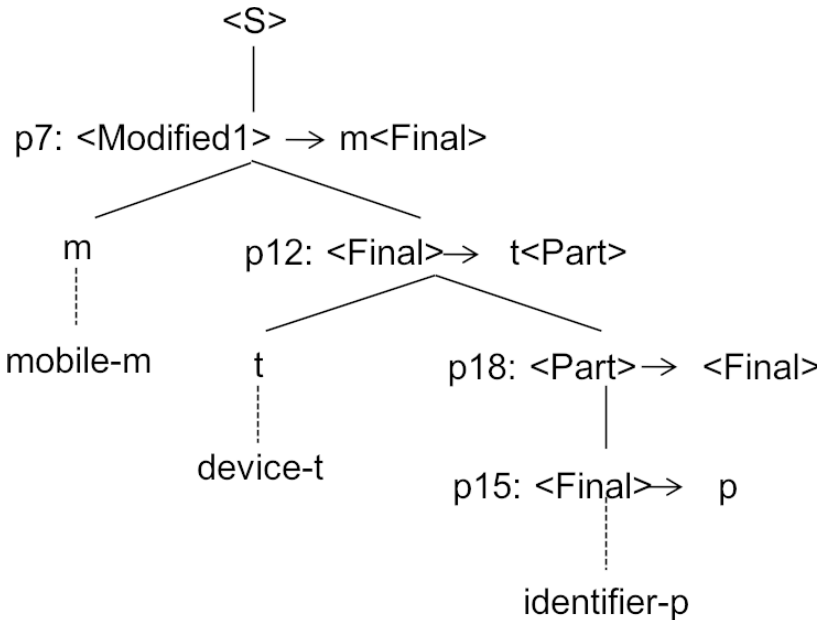


Figure 7.3: Parse Tree for “mobile device identifier” with Tag Sequence “mtp”

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” and “device information”).

Figure 7.3 shows the parse tree for the phrase “mobile device identifier” with type sequence mtp. The tokenized tagged words are presented as “mobile-m”, “device-t”, and “identifier-p” and are read using a lexical analyzer. Next, we discuss how these productions are extended with semantic attachments to infer ontological relationships.

7.1.4 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 [26, 40]. We adapt this principle to infer information type phrase semantics from its constituent morphological variants by extending the CFG production rules with semantic attachments.

Each production $r \in R, r : \alpha \rightarrow \beta_1 ... \beta_n$ is associated with a semantic rule:

$\alpha.sem : \{f(\beta_1.sem, ..., \beta_n.sem)\}$

The semantic attachment $\alpha.sem$ states that 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 $\{\ldots\}$ to the right of the production's syntactic constituents. Table 7.2 presents the productions from the CFG shown in Table 7.1 and their semantic attachments described using λ-calculus. The extended production rules formalize heuristics (see section 5) identified in our prior qualitative study of manually constructed ontology [38]. Next, we introduce λ-calculus before presenting an example in which 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 and constants by using two expression forming operations, called application and abstraction [35].

Here we list all the functions used in the semantic attachments. $WordOf(y)$ is a unary function that maps a non-terminal to its tagged phrase sequence. For example, $WordOf$(Final) returns "device identifier-tp" in the parse tree presented in Figure 7.3. In this example, Final refers to the left-side non-terminal of Modifier1.

$Concat(y, z)$ is a binary function and is used to concatenate two tagged phrase sequences, for example $Concat$(mobile-m, information-x) produces "mobile information-mx" from the tagged phrases "mobile-m" and "information-x." $SubVariant(y)$ is a higher-order function that accepts other functions like $Concat$ as an argument. This function returns a list of variants that can be constructed using the input argument. For example, $SubVariant$(mobile device identifier-mtp) returns the following list of variants: [mobile device identifier-mtp, device identifier-mtp, identifier-p].

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

Table 7.2: Semantic Attachments for the Context-Free Grammar

	Production	Semantic Attachments	Line
p1	<S>→<Modified1>	{Modified1.sem}	1
p2	<S>→<Modified2>	{Modified2.sem}	1
p3	<S>→<Final>	{Final.sem}	1
p4	<S>→ x	{λz. IF(IsInfo z) [] (Eqv(z, information-x))}	1

p5	<Modified1$_0$>→m<Modified1$_1$>	{$\lambda y.\lambda m$.Modified1$_1$.sem(Concat(y, m));
		λm.KindOf(WordOf(Modified1$_0$), Concat(m, information-x));
		KindOf(WordOf(Modified1$_0$),
		WordOf(Modified1$_1$))}
			1
			2
			3

p6	<Modified1>→m<Modified2>	{$\lambda y.\lambda m$.Modified2.sem(Concat(y, m));
		λm.KindOf(WordOf(Modified1), Concat(m, information-x));
		KindOf(WordOf(Modified1),
		WordOf(Modified2)}
			1
			2
			3

p7	<Modified1>→m<Final>	{$\lambda y.\lambda m$.Final.sem(Concat(y, m));
		λm.KindOf(WordOf(Modified1), Concat(m, information-x));
		KindOf(WordOf(Modified1), WordOf(Final))}
			1
			2
			3

p8	<Modified1>→ mx	{$\lambda m.\lambda z$. KindOf(Concat(m, z), z)); λz.IF(IsInfo z) [] (Eqv(z, information-x))}	1 2

p9	<Modified2>→a<Final>	{$\lambda y.\lambda a$. Final.sem(Concat(y, a));
		λa.KindOf(WordOf(Modified2), Concat(a, information-x));
		KindOf(WordOf(Modified2), WordOf(Final));
		λa.Eqv(a, Concat(a, information-x))}
			1
			2
			3

p10	<Modified2>→e<Final>	{$\lambda y.\lambda e$. Final.sem(Concat(y, e));
		λe.PartOf(WordOf(Modified2), e);
		KindOf(WordOf(Modified2), WordOf(Final));
		λe.Eqv(e, Concat(e, information-x))}
			1
			2
			3
			4

p11	<Modified2>→a<Info>	{$\lambda y.\lambda a$. Info.sem(Concat(y, a));
		KindOf(WordOf(Modified2), WordOf(Info));
		λa.Eqv(a, Concat(a, information-x))}
			1
			2
			3

p12	<Final>→t<Part>	{$\lambda y.\lambda t$. Part.Sem(Concat(y, t));
		KindOf(WordOf(Final), WordOf(Part));
		Map(λz.PartOf(Concat(z, WordOf(Part)),z))$\lambda y.\lambda$ t.
		SubVariant(Concat(y, t))}
			1
			2
			3

p13	<Final>→t<Info>	{$\lambda y.\lambda t$. Info.sem(Concat(y, t));
		KindOf(WordOf(Final), WordOf(Info));
		λt.Eqv(t,Concat(t, information-x))}
			1
			2
			3

p14	<Final>→e<Info>	{$\lambda y.\lambda e$. Info.sem(Concat(y, e));
		KindOf(WordOf(Final), WordOf(Info));
		λe.Eqv(e,Concat(e, information-x))}
			1
			2
			3

p15	<Final>→p	{(Map($\lambda p.\lambda z$.PartOf(p, z)))λy.SubVariant(y); $\lambda y.\lambda p$.PartOf(Concat(y,p),y)}	1 2
p16	<Part>→<Modified1>	{λy. Modified1.sem(y)}	1
p17	<Part>→<Modified2>	{λy.Modified2.sem(y)}	1
p18	<Part>→<Final>	{λy.Final.sem(y)}	1

p19	<Info>→x	{λz. IF(IsInfo z) [] (Eqv(z, information-x));
		$\lambda z.\lambda y$. IF(IsInfo z) [] (Eqv(Concat(y, z), Concat(y,
		information-x))) }
			1
			2

p20	<Info>→ ϵ	{}	1

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

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

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

We now describe step 3 (from Figure 7.1) using the tagged information type "mobile device identifier-mtp". In step 3, the tagged information type is first parsed using the grammar in Table 7.1 and its semantics are computed by visiting the nodes of the parse tree in Figure 7.3 and applying the corresponding semantic attachments from Table 7.2 during a single-pass, top-down parse. Following this order, the semantics of production rule p7 is mapped to the following λ-expressions, where l in p7.l refers to line l in Table 7.2:

p7.1 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 [40]. 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.

p7.2 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.

p7.3 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")].

In the next step, we analyze the semantics of production rule p12 that are presented using three λ-expressions:

p12.1 represents a λ-expression to concatenate tagged phrases associated with the inherited variable y and variable t which is read through the lexical analyzer, and passes the concatenation result ("mobile device-mt") to direct descendants of this node.

p12.2 represents the application of $KindOf$ function on "device identifier-tp" and "identifier-p", generating a hypernymy relationship between these two tagged phrases as a single element list.

p12.3 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-tp"), 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-tp." 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 the 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 the words in "mobile device identifier-mtp" helps us consider all the possible semantics associated with an ambiguous phrase.

Moving to the next node (p18) in the parse tree, the semantic attachment **p18.1** 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 p15 as the last node visited in the parse tree is mapped to the following attachments:

p15.1 is the application of Map to a variant list constructed from a λ-abstraction. This abstraction is reduced to SubVariant(“mobile device-mt”), which returns the following variant list as a result: [“mobile device-mt”, “device-t”]. Finally, Map applies $PartOf$ function on all the elements of this list resulting in [PartOf(“identifier-p”, “mobile device-tp”), PartOf(“identifier-tp”, “device-t”)].

p15.2 represents an abstraction that reduces to [PartOf(“mobile device identifier-mtp”, “mobile device-mt”)].

All the above production rules processing the tagged information type “mobile device identifier-mtp” yield a collection of candidate semantic relationships contained in multiple lists, in addition to the inferred morphological variants of the input information type. As the final procedure in step 3, we merge these lists and translate the relations into corresponding description logic axioms, which are then added to the output ontology. One might argue that variants such as “mobile information-mx” generated using this method are not valid phrases. However, the similar rules can be used to generate valid variants, such as “anonymous information-mx” and “demographic information-mx” from “anonymous demographic information.” Therefore, the variants and relationships are considered candidates.

7.2 Evaluation and Results

We evaluate the ontology construction method by answering the following research questions:

RQ1: How much, and to what extent, does the context-free grammar with semantic attachments cover the relationships between information type pairs in Lexicon L_1?

RQ2: Which semantic relations are missed by the method in comparison with the ground truth ontology?

RQ3: Considering a generative treatment of morphology, to what extent does the grammar generate tag sequences that cannot be analyzed using the previously proposed approaches?

RQ4: What level of effort is required to maintain the method for each new lexicon addition?

RQ5: How reliable is the method with respect to a new lexicon addition?

Research questions RQ1, RQ2, and RQ3 evaluate the ontology construction method using lexicon L_1, discussed in Section 7.2.1. Research questions RQ4 and RQ5 evaluate the generalization and coverage of our method using lexicon L_2, discussed in Section 7.2.2.

7.2.1 Evaluation using Lexicon L_1

In this section, we evaluate the ontology construction method using lexicon L_1 to answer the research questions RQ1, RQ2, and RQ3. This lexicon contains 356 information types which were used to develop the context-free grammar (CFG) in Section 7.1.3. We acquired the reduced and tagged information types in L_1 through this link [1]. Given 335 reduced tagged information types, the CFG and semantic attachments in step 4 (see Figure 7.1) yield 4,593 ontological relations that share at least one common word. We plan to publish the inferred relations publicly both in text and OWL format.

We require a ground truth (GT) ontology containing the relationships (hypernymy, meronymy, or synonymy) between information types in lexicon L_1 to evaluate the accuracy of the inferred relations to answer RQ1. We acquired the results of a study published by Hosseini et al. [37] [2] and followed their approach to construct the GT. This study contains 2,253 information type pairs which is the result of pairing all the information types that share at least one word in the reduced version of lexicon L_1 (based on step 1). This study contains the relationships assigned to each information type pair by 30 human subjects per pair, which are called participant preferences. The participants in this study were recruited from Amazon Mechanical Turk and had completed over 5,000 HITs, had an approval rating of at least 97%, and were located within the United States [37].

Due to the diversity of participant experiences, which allows participants to perceive different

[1] http://gaius.isri.cmu.edu/dataset/plat17/

[2] http://gaius.isri.cmu.edu/dataset/plat17/preferences.csv

interpretations or senses of phrases, different participants can assign different semantic relations to the same phrase pair, e.g., “mac” can refer to both a MAC address for Ethernet-based routing, and a kind of computer sold by the computer manufacturer Apple. In another example, “email” can refer to three different senses: a service or program for sending messages; a message to be sent via the SMTP protocol; or to a person’s email address, which is the recipient address of an email message. Therefore, participants may conclude “email address” is a part of “email”, or is equivalent to “email” which are both valid interpretations. To avoid excluding valid interpretations, we [37] chose to build a multi-viewpoint GT that accepts multiple, competing interpretations. To this end, we define valid interpretations for a phrase pair to be those interpretations where the observed number of responses per category exceeds the expected number of responses in a Chi-square test, where $p < 0.05$ for the entire survey result set. This threshold means that there is at least a 95% chance that the elicited response counts are different than the expected counts [37]. The expected response counts for a semantic relationship are based on how frequently participants chose that relationship across all participant comparisons. Finally, we constructed a multi-viewpoint GT as follows: for each surveyed pair, we add an axiom to the GT for a relation category, if the number of participant responses is greater than or equal to the expected Chi-square frequency; except, if the number of unrelated responses exceeds the expected Chi-square frequency, then we do not add any axioms.

We compared the inferred relationships with the relationships in the GT. A semantic relationship is a true positive (TP), if it is logically entailed by GT, otherwise, that relationship is a false positive (FP). Overall, the ontology fragments inferred using our approach logically entail 980/2,253 of hypernyms, meronyms, and synonyms in the GT. We use logical entailment to identify TPs, because subsumption is transitive and whether a concept is a hypernym of another concept may rely on the transitive closure of that concept’s class relationships in the GT.

For all information type pairs with valid interpretations that do not match an inferred semantic relationship, we count these as false negatives (FN). We found 466/2,253 of the related pairs in the GT that cannot be logically entailed in the ontology fragments inferred through our method.

Table 7.3: Performance Measures for Lexicon L_1

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

We computed Precision(Prec.) = TP/(TP+FP) and Recall(Rec.) = TP/(TP+FN) for the ontology construction method using CFG and semantic attachments, presented in Table 7.3. We also compare the results of our method with our previously proposed ontology construction method using 26 regular expression patterns (see Section 6.1) [37]. Our model outperforms the 26 regular expression patterns, by decreasing the number of FNs and improving the recall.

RQ2 concerns the type of relationships that cannot be inferred using the CFG and it's semantic attachments. To answer this question, we open coded the 466 FNs and identified four codes that explain the reasons that our method could not infer the relationships:

(1) *Tacit Knowledge:* The relationship requires tacit knowledge to be inferred and may not be inferred using syntax analysis of phrases, alone. For example, the hypernymy relationship 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 is unavailable in our method. We identified 404/466 of the FNs fall into this category.

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

(3) *Modifier Suppression:* Participants may ignore modifier roles in a phrase and thus prefer an equivalent relationship between a pair of phrases. For example, "actual location" and "approximate location" are identified equivalent in the GT ontology. We also reported this phenomenon in our

early work discussed in Chapter 6 [37]. We identified 34/466 phrase pairs and their relationships that fall into this category.

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

During second-cycle coding of relationships in Tacit Knowledge category, we observed a potential explanation for why individuals prefer a relationship that differs from our method results. During step 2, we classified the terms in "application software" using the sequence "tt," which assumes each term is a thing, which is used in our approach to entail that "software" is part of an "application." However, we believe that participants recognize that "application software" is a single thing. We also believe this explanation applies to 20 phrases and 69 semantic relations in the GT. Other examples of this kind include "web page" in "web page visited" and "geographic location." We revised the tag sequences for these 20 phrases and inferred ontological relations based on the revision. Applying our method on the set of revised tagged information types results in additional 74 FNs compared to the original tagged information types. For example, the method cannot infer the relationships between the following pairs: (application software, software information), (page view order, web page), and (geographic information, geographic location information) due to tag changes. Our ontology construction method solely relies on syntax analysis and tag sequences of the information types. Therefore, semantic ambiguity in tokenization and tagging can result in changes in the inferred relationships which can be considered as one of the shortcomings of our method.

Considering a generative treatment of morphology [20], a grammar needs to account for active word formation [2]. To answer RQ3, we evaluate the formation of noun phrases using the CFG introduced in Section 7.1.3 by generating all possible tag sequences of varying lengths from 1-4 tokens. The generation process begins with a sequence consisting of the start symbol, followed by each reachable production on the left-hand side, replacing the start symbol with the right-hand side of the production, and then we repeat this process of selecting non-terminals in the sequence and replacing them with the right-hand side of the corresponding production, until all non-terminals are resolved and replaced by terminals. This approach yields 129 unique tag sequences, which are restricted to contain only two consequent modifiers (e.g., we won't allow a sequence "mmmt" to be produced). We analyzed the generated sequences using the 26 regular expressions discussed in Chapter 6 [37]. The results indicate that only 47/129 of generated tag sequences match the sequences created for phrases in the L_1 lexicon. This means that the CFG is potentially more expressive than our earlier ontology construction method using 26 regular expressions. Also, additional evaluation is needed to determine if the generated sequences are valid or would simply be unused.

7.2.2 Evaluation using Lexicon L_2

The research questions RQ4 and RQ5 ask about the level of effort to maintain the method, and the method's reliability. We pre-processed 1,853 information types in lexicon L_2 using the strategies mentioned in Section 7.1.1, yielding 1,693 information types. In the four steps presented in Figure 7.1, only step 2 involves manual effort for semantic tagging. During this step, two analysts individually assigned semantic role tags to information types in L_2. We calculated the inter-rater agreement for the assigned tags using Fleiss' Kappa co-efficient, which is a chance-corrected measure of agreement between two or more raters on a nominal scale [25]. The comparison resulted in 518 disagreements with Kappa = 0.704. Next, the analysts reconciled their differences, and Kappa increased to 0.917. We used the reconciled tagged results from one of the analysts as the input for the next step. During this manual step, the required time is linear in the size of the lexicon compared to the manual ontology construction method discussed in Chapter 5 [38], in which the effort required to construct the ontology is $(n \times (n-1))/2$ pairwise comparisons for n information types.

Next, we constructed information type pairs which share at least one word, yielding 1,466,328 pairs. Due to the large number of pairs, we sampled the pairs by creating strata that represent

Table 7.4: Performance Measures: Lexicon L_2

TPs	FPs	FNs	TNs	Prec.	Rec.
1,686	3	156	438	0.99	0.91

comparisons between tag sequences as follows:

Step 1: Each information type pair is mapped to their respective tag sequence pair, e.g., pair (mobile device, device name) is mapped to (mt, tp), yielding 974 unique tag sequence pairs, which we call the strata.

Step 2: Proportional stratified sampling is used to draw at least 2,000 samples from all strata with layer sizes ranging from 1 to 490. The wide range in layer sizes implies unbalanced strata; e.g., strata that contain 1-3 pairs when divided by the total number of information type pairs yields zero. Therefore, we guarantee that all information type pairs in a strata of size one are selected to ensure each strata is covered. Next, for strata of size two and three, one random information type pair is selected. For the remaining strata with sizes greater than three, sample sizes are proportional to the strata size, which yields 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 as follows. First, we create a *bag-of-lemmas* by obtaining word lemmas in the already selected pairs. Next, in each stratum, the pairs with the least common lemmas with the bag-of-lemmas are selected. We update the bag-of-lemmas after each selection by adding the lemmas of the selected information type pairs. This strategy insures that the information type pairs with the lower similarity measure is selected, resulting in a broader variety of words in the sampled set. Moreover, we ensure that each tag sequence is represented by at least one sampled item, and that sequences with a larger number of examples are proportionally represented by a larger portion of the sample. Using the initial sample size of 2,000, we captured 2,283 samples from 1,466,328 phrase pairs. The original sample size differs from the final one due to our strategy for ensuring at least one sampled pair for each strata. Our samples contain 1,138 unique information types from Lexicon L_2.

To address RQ5 on method reliability, we require a ground truth for relations between the information types within the sampled pairs. For this reason, we published a survey following the method described in Section 6.3 [37]. The survey asks subjects to choose a relation for pair (A, B) from one of the following six options:

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.”

For this survey, we recruited 30 qualified Amazon Mechanical Turk participants following the criteria mentioned in Section 7.2.1. Using the survey results and the approach mentioned in Section 7.2.1, a multi-viewpoint ground truth (GT) was constructed. We plan to publish the survey results and the GT publicly. We measure the number of true positives (TP), false positives (FP), and false negatives (FN) by comparing the semantic relations with the multi-view GT to compute Precision (Prec.) and Recall (Rec.), which are presented in Table 7.4. Our method yields 21,745 total semantic relations from the sampled information types, which we plan to publish publicly both in text and OWL format. Overall, the method correctly identifies 1,686/2,283 of semantic relations in the GT ontology.

7.3 Conclusion and Future Work

In this chapter, we introduced a method to infer semantic relations between information types in privacy policies and their morphological variants based on a context-free grammar and semantic attachments. This method is constructed based on grounded analysis of information types in 50 privacy policies and tested on information types from 30 policies. Our method shows an improvement in reducing the number of false negatives, the time, and effort required to infer semantic relations, compared to our previously proposed methods by formally representing the information types.

In the next chapter, we discuss our neural network classification model to infer semantic relations that are independent of syntax and purely rely on tacit knowledge, such as hypernymy relation between “phone” and “mobile device.”

Chapter 8: LEARNING ONTOLOGIES FROM NATURAL LANGUAGE POLICIES

Our prior work to construct a privacy ontology (See Chapter 5) [38] requires comparing information types with every other type in a lexicon and assigning a semantic relationship to each pair using seven heuristics. The required effort for this task is quadratic $m \times n \times (n-1)/2$, where n is the number of information types in the lexicon, and m is the amount of time required to assign a relationship to a pair, estimated at 20 seconds [37]. In addition, a new policy introduces between 11-36 new types that are not encountered in the existing lexicon [6]. Considering app markets contain hundreds of thousands of apps that change daily, we need to automate ontology construction. Thus, we propose to predict candidate relationships between information type pairs. Unlike syntax-based method we proposed in Chapters 6 and 7, our approach relies on tacit knowledge by learning the semantics of words and phrases using word embeddings and convolutional neural network (CNN).

In this chapter, we describe an empirical method to learn and construct a formal ontology from a naïve set of all pairs of information types contained in a lexicon. The contributions of this chapter are three-fold: (1) a novel neural network architecture for learning to predict semantic relationships among information type pairs; (2) a novel method to sample an existing ontology to create a training and testing set that accounts for dependencies among concepts and formal ontological relations; and (3) an empirical evaluation of the neural network on a real dataset. The remainder of this chapter is organized as follows. In Section 8.1, we present the relation classification model to learn an ontology; Section 8.2 describes the experiment designs; the evaluation and results appear in Section 8.3; and finally, in Section 8.4 we present discussion, limitations of our work, and future direction.

8.1 Relation classification Model

Figure 8.1 shows the learning architecture for the relation classification model with a pair of information types as input. Throughout the chapter, we present the information type pair as information-type$_{LHS}$ and information-type$_{RHS}$, e.g., (device information, device ID), where LHS (left-hand side) and RHS (right-hand-side) indicate two predicates in an asymmetric ontological relationship. The input information types in a privacy policy lexicon can be from a single statement in a policy, different sections of a single policy, or completely different policies. Given an information type pair, the Embedding layer first maps the words in an information type to their corresponding word embedding vectors. Second, word embeddings are fed into the Phrase Modeling layer, creating a phrase-level semantic vector for each information type phrase. Third, the Semantic Similarity Calculation compares the direction and distance of the two phrase-level vectors and generates a similarity vector. Finally, the similarity vector is input to Softmax, generating three probabilities corresponding to hypernymy, synonymy, and unrelated. We select the most probable relation for each information type pair.

We now describe these four steps in further detail.

8.1.1 Embedding Layer

Each word in an information type phrase is presented using a pre-trained 200-dimensional vector called a word embedding [5]. To create domain-specific word embeddings, we followed the approach by Harkous et al. [32] and trained the Word2Vec model (see Section 2.6.2) using 77,556 English privacy policies collected from mobile applications on the Google Play Store . Common-purpose word embeddings trained on the English Wikipedia dump [8, 44, 52] or Google News dataset [46] exist, however, previous research has shown improvements on classification accuracy by utilizing domain-specific word embeddings [66].

To obtain the privacy policy corpus to train the Word2Vec model, we crawled the metadata archive for more than 1,402,894 Android apps provided by the PlayDrone project [68] from which 109,933 contained a valid link to a privacy policy. We used the BeautifulSoup library in Python to

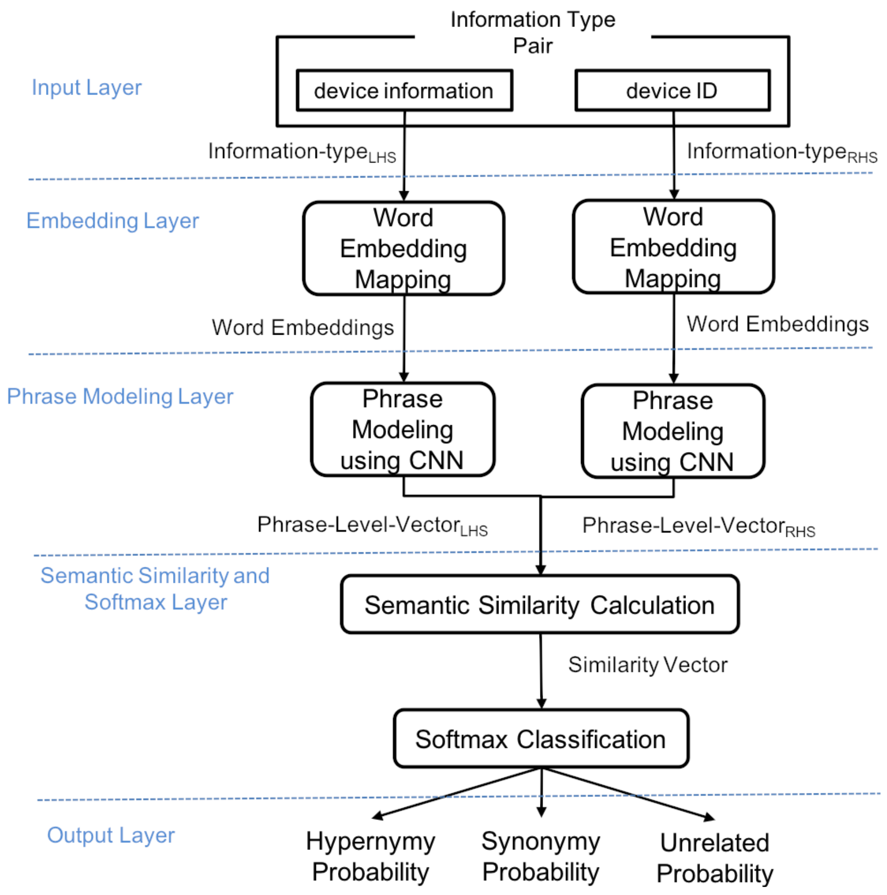


Figure 8.1: Ontology Learning Architecture

extract the text from the HTML files by stripping HTML tags associated with: head, script, URL, navigation, button, and option information. Next, we filtered non-English policy text files, yielding 77,556 privacy policies with the majority of text in English by using the DetectLang library in Python. In the next step, for each privacy policy, we tokenized the sentences and removed all non-English sentences. We also expanded the contractions (e.g., “won’t” is transformed to “will not”), and removed punctuation, numbers, email addresses, URLs, and special characters. Finally, we transformed the remaining characters to lower-case. The resulting pre-processed text was used to train the Word2Vec [46] model.

The trained word embeddings for the words in our privacy policy corpus are stacked in a word embedding matrix, which is used in the mapping process. The Embedding layer maps every word in an input information type phrase from a privacy policy lexicon to its corresponding embedding vector read from the embedding matrix. To this end, we first identify the maximum phrase length t by analyzing the number of words in all information types in the privacy policy lexicon. Next, the information type phrase is padded automatically if the number of words is less than t to reach the maximum length. This approach ensures that all the input information types have the same length. Next, using the word embedding matrix, each word in the padded information type is mapped to its corresponding word embedding vector. If a word cannot be found in the embedding matrix, our approach assigns a 200-dimension vector to the word with its elements randomly generated using the uniform distribution. In the next section, we illustrate how the word embeddings for each padded information type phrase are utilized to generate a phrase-level semantic vector.

8.1.2 Phrase Modeling Layer

In this section, we describe the Phrase Modeling layer (see Figure 8.2) that transforms word embeddings of an input information type to a low dimensional, fixed-sized vector using CNN with three different filter widths [66]. Implementing CNN with multiple filter widths captures local semantics of n-gram of various granularities [65]. In our case, convolutional filters with widths 1, 2, and 3 capture the semantics of uni-grams, bigrams, and trigram, respectively.

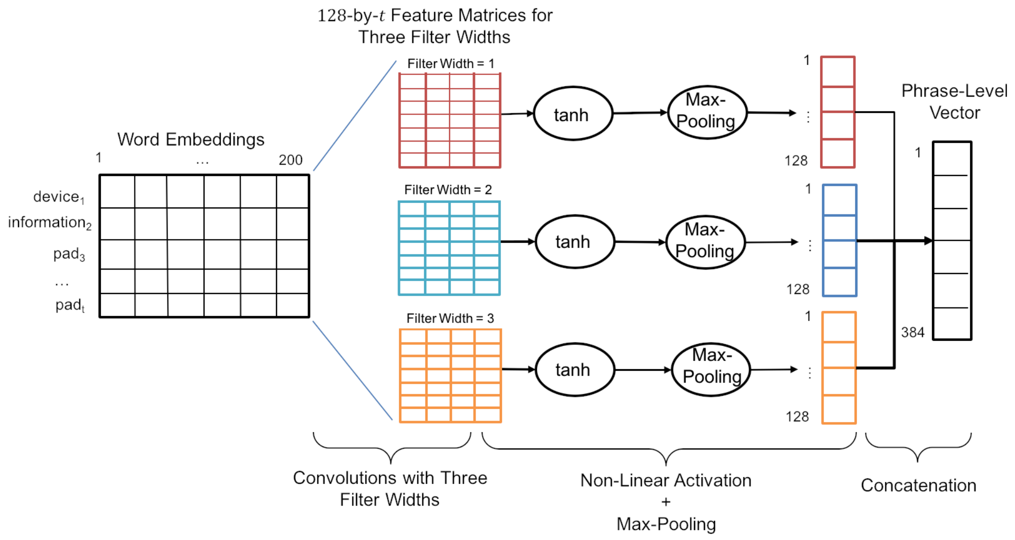


Figure 8.2: Phrase Modeling Layer

We present an example for convolution filter of width $w = 3$ for the padded information type $P : device_1 information_2 pad_3 \dots pad_{t-1} pad_t$ with length t, where t is the maximum phrase length in the privacy policy lexicon as discussed in Section 8.1.1. The words/pads in the information type P are represented as a list of vectors $(x_1, x_2, \dots, x_{t-1}, x_t)$, where $x_i \in R^n$ corresponds to the word embedding of word/pad $i \in P$ and n represents the dimension of word embeddings $n = 200$ (see Section 8.1.1). Our approach automatically assigns a 200-dimension vector with random uniform values for pads and also the words that cannot be found in the embedding matrix. Using embedding vectors and filter width $w = 3$, the phrase is represented as follows: $\{[x_1; x_2; x_3], \dots, [x_{t-2}; x_{t-1}; x_t]\}$, where ";" shows vertical vector concatenations. In general, the result of this module is a matrix $X \in R^{n_0 \times t}$, where $n_0 = w \times n$. To convolve all the features in X, we process X using the linear transformation in Equation (8.1).

$$Z = W_1 X \tag{8.1}$$

where $W_1 \in \mathbb{R}^{n_1 \times n_0}$ is the linear transformation matrix and n_1 is a hyper-parameter representing the number of filters, which we set as 128 in our approach. The result of linear transformation is shown as $Z \in R^{n_1 \times t}$, which is dependent on t, the maximum phrase length in the privacy policy lexicon.

We further apply hyperbolic tangent (tanh) as a non-linear activation function, see Equation (8.2) on the result of the linear transformation. To determine the useful features, we apply a maximum pooling on h. The result of this process is a feature vector of size n_1 which is independent of the phrase length.

$$h = tanh(Z) \tag{8.2}$$

After applying the Phrase Modeling layer to the input information type with three different convolution filter widths, we retrieve three high-level feature vectors of size n_1 as shown in Figure 8.2. Finally, we concatenate these three feature vectors to create a single vector representing the phrase-level semantics. We follow this approach to generate two phrase-level vectors for both information-type$_{LHS}$ and information-type$_{RHS}$. These two vectors are compared in the Semantic Similarity Calculation layer, which we discuss next.

8.1.3 Semantic Similarity and Softmax

The Semantic Similarity Calculation layer compares the input phrase-level vectors from the Phrase Modeling layer. We adopt the structure proposed by Tai et al., where the direction and distance of the two input vectors are compared using the following equations [64]. Two vectors PLV_{LHS} and PLV_{RHS} refer to Phrase-Level-Vector$_{LHS}$ and Phrase-Level-Vector$_{RHS}$ in Figure 8.1, which are shortened for simplicity in the equations.

$$dir = PLV_{LHS} \odot PLV_{RHS} \tag{8.3}$$

$$Dis = |PLV_{LHS} - PLV_{RHS}| \tag{8.4}$$

$$sim = \sigma(Wdir + Udir + b) \tag{8.5}$$

Equation (8.3) compares the direction of two semantic vectors PLV_{LHS} and PLV_RHS for each dimension using the point-wise multiplication operator. For calculating the distance between PLV_LHS and PLV_{RHS}, we utilize the absolute vector subtraction presented in Equation (8.4). To integrate the results of Equation (8.3) and Equation (8.4) on PLV_LHS and PLV_RHS, we use a hidden sigmoid layer presented in Equation (8.5). The similarity vector as the output of the function is then sent to a Softmax classifier as shown in Equation (8.6) to predict the probabilities of hypernymy, synonymy, and unrelated. We select the prediction with the highest probability as the relationship between information-type$_{LHS}$ and information-type$_{RHS}$. Next, we discuss the loss function used to train the relation classifier.

$$P_{relation} = softmax(W_p sim + b_p) \tag{8.6}$$

8.1.4 Loss function

We use weighted cross-entropy loss to measure the performance of the relation classifier with respect to the predicted probability $P_{relation}$, which has been normalized with Softmax, and the actual label (hypernym, synonymy, unrelated). Cross-entropy loss increases as the predicted probability $P_{relation}$ diverges from the actual label. We use weights to account for imbalance in ontological relations, i.e., the number of unrelated pairs is several orders of magnitude larger than all other relations combined. The weights are calculated using the frequency of each relation's presence in the ontology as a simple ratio: unrealetd/total and ((hypernymy+synonymy))/total. The unrelated ratio is applied to the hypernymy and synonymy classes, as determined by the actual label, to give more weight when determining the loss, and the hypernymy and synonymy ratio is applied to the unrelated class to give less weight when determining the loss. The loss function is defined in Equation (8.7).

$$loss = -w^i \sum_{i \in T} y^i ln(P^i_{relation}) \tag{8.7}$$

where T is the training pairs, and w^i, y^i, and $P^i_{relation}$ are the weight, actual label, and predicted probability, respectively, for the i^{th} information type pair in the training set.

The training involves sending the derivative of the loss function through back-propagation to update the network parameters using the stochastic gradient descent method. Each epoch iterates over all the training data, which are divided into multiple batches. After processing each batch of training data within an epoch, the parameters are updated based on the gradient of the loss function and another hyper-parameter called the learning rate. This hyper-parameter determines how fast or slow the network should move towards the optimal solution. The hyper-parameters, including learning rate, are defined in Sections 8.2.4 and 8.2.5. The training process stops when the loss value is sufficiently small or fails to decrease [30].

8.2 Experiment Designs

In this section, we first describe our motivation for evaluating the relation classification model, before describing our experimental designs.

The ability to predict an ontology relationship is a multi-class classification problem, in which given an ordered information type pair, we are interested in whether the first item in the pair is a hyponym, synonym or in another type of relation, including unrelated. To this end, we consider two views about how the classification model learns to predict these relationships: (1) each relationship is independent, and the network has learned direct relationships among concepts ignoring the transitive closure of hypernymy; or (2) relationships can be dependent on one another, and the network has learned a partial semantic representation that is some subset of the transitive closure of hypernymy. For example, we assume that (Android ID, mobile device ID) are related through hypernymy relationship. Similarly, hypernymy holds for (mobile device ID, device identifier). We assume the classification model learns semantic relationships among these words based on how they are used in policy sentences. Under view one, we train the classification model to only learn these two relationships from the embeddings trained on the policies. Under view two, however, we further train the model to learn relationships inferred through transitivity, which includes the hypernymy relationship for (Android ID, device identifier). We hypothesize that this additional training generalizes to improve the classification of hypernymy, because more abstract hypernyms can be used to group semantically similar, but not directly related concepts.

We conducted two experiments to evaluate the relation classification model described in Section 8.1. In experiment 1, we evaluate the model's ability to classify whether an information type pair is a direct hypernymy, synonymy, or otherwise. Experiment 2 differs from experiment 1 by considering entailed hypernymy relations, which include direct and indirect hyponyms in a single class, and thus we evaluate the model's ability to classify information type pairs into one of three classes: hypernymy (direct and indirect), synonymy, or otherwise.

In this section, we first discuss the unique challenge of learning with ontologies. Second, we introduce our ground-truth ontology. Third, we introduce a method to generate an early version of the ontology which serves as the training ontology for our model. Finally, we discuss the experiments.

8.2.1 Learning with Ontology

In traditional machine learning, each data point in a data frame is independent and thus data points can be randomly divided into training and testing data. Models are fitted to training data, and then classifications or predictions are evaluated on testing data. When predicting extensions to an ontology, however, the relationships between ontology classes are not independent: the relationships in hypernymy are transitive, and thus removing a relationship between a superordinate and subordinate concept can lead to misclassification between the subordinate concept and its ancestor concepts. To address this challenge, each ontology is treated as a versioned dataset, wherein later versions contain new information types dependent (via relations) on types found in earlier versions. To learn a later version, we train the relation classification model on an earlier ontology version. The earlier version is constructed by randomly eliminating information types from the later version, while repairing the earlier version so that all entailments of the early version are contained in the entailment of any subsequent version (i.e., for each future version, the entailment is monotonically increasing). Therefore, we train the relation classification model on an early version of

Table 8.1: Examples of Information Types in Platform Information Lexicon L

Information Type	Frequency
IP address	41
browser type	21
Location	11
Operating system type	5
Geo-location	4
Ads clicked	2
Media access control	2
Adverstising ID	1
Device brand	1

the ontology (training ontology), and we validate the model on the later version (testing ontology). We now introduce the ontology used to evaluate our method followed by the steps to generate an early version for training purposes.

8.2.2 Platform Ontology

As our ground-truth, we utilize the platform ontology (see Section 5.2) that is manually built from the platform information lexicon (see Section 4.2), which we call L. This lexicon was extracted from 50 privacy policies [60]. Lexicon L contains phrases that correspond to platform information types, defined as "any information that the app or another party accesses through the mobile platform that is not unique to the app." Each information type in L has a frequency, or number of times the type appeared in annotations of the 50 policies. Table 8.1 shows example types and their frequencies in L.

As mentioned in Chapter 5, we manually constructed the platform ontology from L by applying seven heuristics that were identified through grounded analysis of five privacy policies [38]. The platform ontology contains 367 information types, which are used to comprise 1583 hypernymy and 310 synonymy relationships between pairs of information types.

Formally, the platform ontology is a knowledge base KB expressed using FL_0, a sublanguage of the Attribute Language (AL) in Description Logic (DL). A DL knowledge base KB is comprised of two components, the TBox and the ABox [3]. 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. The platform ontology knowledge base KB only contains terminology, which we call the TBox T.

The semantics of 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 set $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$. The TBox T also contains terminological axioms that relate concepts to each other in the form of subsumption and equivalence, which we use to formalize hypernymy and synonymy, respectively. A concept C is subsumed by a concept D, written $T \models C \sqsubseteq D$, if $C^{\mathcal{I}} \sqsubseteq D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$ that satisfy the TBox T. The concept C is equivalent to a concept D, written $T \models C \equiv D$, if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$ that satisfy the TBox T. Axioms of the first kind ($C \sqsubseteq D$) are called inclusions, where axioms of the second kind ($C \equiv D$) are called equalities [3]. Note that the equalities $C \equiv D$ can be rewritten as two inclusion axioms $C \sqsubseteq D$ and $D \sqsubseteq C$ [63]. Using this formal representation, we now describe our method to construct an early version of platform ontology.

8.2.3 Training Ontology

The procedure to create an early version of an ontology is analogous to sampling from a graph, wherein concepts (nodes) are related via axioms (edges). Thus, we first briefly introduce traditional graph sampling goals, after which we introduce our sampling goal and corresponding method.

Graph sampling is the problem of creating a small sample graph that has similar properties as the target graph [43]. Scale-down sampling and back-in-time sampling describe two common goals in graph sampling [43]. In scale-down sampling, the goal is to create a sample S on n' nodes from a static graph G containing n snodes, where $n' \ll n$. Sample S has similar properties as graph G, such as degree distribution, clustering coefficient distribution, and hop-plot. Back-in-time sampling corresponds to traveling back in time and trying to mimic past versions of graph G [43]. Let $G_{n'}$ denote the graph G at some point in time, when it had exactly n' nodes. The goal is to find a sample S on n' nodes with similar properties as graph $G_{n'}$, i.e., when graph G was the size of S. Sampling methods for scale-down and back-in-time sampling are summarized in three main groups: (1) methods based on randomly selecting nodes; (2) methods that select edges randomly; and (3) exploration techniques that simulate random walks on a target graph.

In our approach, we are training a supervised machine learning algorithm to predict future versions of an ontology, and thus our goal is to create a training ontology T' that can be used to construct a training set for our learning task from a target ontology T with n concepts. We call this sampling goal version sampling, which has the following two properties:

P1. The training ontology T' follows the scale-down sampling goal, where the number of sampled concepts $n' \ll n$.

P2. The training ontology T' is an early version of the target ontology T with n' concepts, similar to back-in-time sampling.

Our sampling goal differs from the traditional graph sam-pling goals for two reasons: (1) scale-down sampling can yield disconnected graphs and node reachability is not the main concern, whereas transitivity is essential to subsumption inference and ensuring that the entailment of future ontology ver-sions is monotonically increasing; (2) back-in-time sampling aims to sample on an early version of the graph, whereas our aim is to use the early version of the graph as the sample.

Version sampling is comprised of two algorithms: (1) a weighted random sampling algorithm to identify candidate concepts to remove from the target ontology; and (2) the remove-and-repair algorithm that repairs the modified target ontology after removing concepts from the target T to yield the early version ontology T'.

Weighted Random Sampling

Concepts in the platform ontology occurred across 50 privacy policies according to the frequencies recorded in the lexicon L. When choosing concepts to remove from the target ontology to yield an earlier version, one must consider the likelihood that the concept would be seen in a policy, before it was added to the ontology during construction. For this reason, we introduce a probability proportional to size sampling algorithm that is a weighted random sampling method [51]. This algorithm iteratively takes an information type from lexicon L with inclusion probability inversely proportional to the information type frequency in L. The algorithm terminates after sampling n'' information types from L.

Weighted Random Sampling Algorithm:

Input: Lexicon L with n information types and their frequencies

Output: Sampled lexicon L' with n'' sampled information types

```
1: While sample-size $< n''$
2:     Randomly select an information type $i \in L - L'$
3:     Let $F_i$ be the frequency of information type $i$ in $L$
4:     If $F_i \leq 0$ then
5:         Select information type $i$ and insert it to $L'$
6:     Else
7:         Decrement $F_i$ by one
8: End-While
```

The weighted random sampling algorithm ensures the selection of information types with lower frequency. Elimination of sampled information types from platform ontology results in an ontology T' that is more likely to contain information types with higher frequency, hence higher probability to appear in the privacy policies.

Early Version Repair Algorithm

After the n'' information types have been selected using weighted random sampling, we apply the remove-and-repair algorithm to the target ontology T to yield the early version ontology T', which contains $n' = n - n''$ concepts. In this algorithm, let $f(x) : L' \rightarrow concepts$ in T.

Remove-and-Repair Algorithm:

Input: Sampled lexicon L', Platform ontology as TBox T

Output: Training ontology as TBox T' with n' concepts.

```
1: Create a copy of TBox $T$, called TBox $T'$
```

```
2: For information type $i \in L'$
3:     Take all inclusion axioms of form $C_j \sqsubseteq f(i)$ in $T'$
4:     Take all inclusion axioms of form $f(i) \sqsubseteq P_k$ in $T'$
5:     Create axioms $C_j \sqsubseteq P_k$ and add it to $T'$
6:     Take all equalities of form $f(i) \equiv E_m$ in $T'$
7:     Create axioms $E_m \sqsubseteq P_k$ and add it to $T'$
8:     Omit concept $f(i)$ and all inclusion and equality axioms containing $f(i)$ in $T'$
9: End-For
```

For both experiments, we apply the version sampling algorithm with $n'' = 100$ on the platform ontology TBox T which contains $n = 367$ concepts. The generated training ontology T' contains $n' = 267$ concepts, thus version sampling method satisfies property $P1 : n' \ll n$. In addition, T' is a consequence of T, denoted as for every axiom $t \in T'$, it is true that $T \models t$ for all interpretations T that satisfy the TBox T [63].

We illustrate the version sampling algorithm using a hypothetical ontology in Figure 8.3 that is created using Web Ontology Language (OWL) in Protégé[1], an open-source ontology editor. Every concept in OWL is a sub-class of the class owl:Thing, which is the top concept in DL. In Figure 8.3, concepts C-2 and C-3 are equivalent and this equality is shown using two inclusions. Herein, let $L' = \{$C-3, C-5$\}$. Both L' and the ontology in Figure 8.3 are the input to the version sampling algorithm. First, we create a copy of the ontology called T'.

For concept C-3, we list all the inclusion axioms of form $C_j \sqsubseteq$ C-3 representing the sub-classes of C-3. This list only contains C-4, which we show in the form of a singleton set: $\{$C-4$\}$. Next, we list all the inclusion axioms of the form C-3 $\sqsubseteq P_k$, stating the super-classes of C-3. This list only contains $\{$C-1$\}$ as the super-class of C-3. We now create a new inclusion axiom C-4 $\sqsubseteq$ C-1 and add it to the TBox T'. Next, we list all the equalities for concept C-3, which only includes $\{$C-2$\}$. In line 8 of the version sampling algorithm, we create a new inclusion axiom C-2 $\sqsubseteq$ C-1 and add it to T'. Finally, we omit concept C-3 and axioms C-3 $\sqsubseteq$ C-1 and C-3 $\equiv$ C-2 from the new ontology.

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

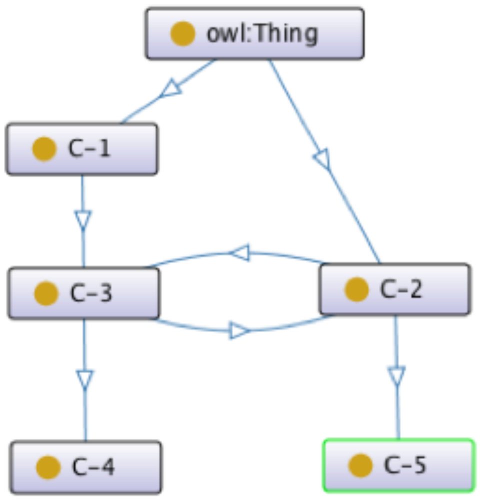


Figure 8.3: An Example Ontology

For the second iteration, we repeat lines 3-8 of the algorithm for concept C-5. Since C-5 has no sub or equivalent classes in the target ontology, there is no need to repair the ontology by adding additional inclusion axioms. Therefore, the algorithm omits C-5 from T' and terminates.

The training ontology generated through our method contains 1,026 hypernyms and 183 synonyms. We now describe our two experiments. Training ontology and platform ontology (i.e., our testing ontolo-gy) are static artifacts in both experiments.

8.2.4 Experiment 1

In experiment 1, we aim to classify whether a new information type pair describes a direct hypernymy relationship, a synonymy, or otherwise. To this end, we define direct hypernymy for concepts C, D if their relation satisfies the following three criteria: (1) $C \sqsubseteq D$; (2) there exists no concept E such that $C \sqsubseteq E$ and $E \sqsubseteq D$; and (3) $C \not\equiv D$. We define synonymy relationship for concepts C, D if $C \equiv D$. If a pair is related in any way other than a direct hypernymy or synonymy relationship, we classify this relationship unrelated. Using this definition, we identify direct hypernyms, synonyms, and unrelated pairs in the training ontology as training-set$_1$ for this experiment. Similarly, we identify direct hypernyms, synonyms, and unrelated pairs in the testing ontology as testing-set$_1$ used to evaluate the model. Both datasets are available online .Table 8.2 presents the

Table 8.2: Experiment 1: Number of Hypernym, Synonym, and Unrelated Pairs in Training and Testing Sets

	Direct Hypernymy	Synonymy	Unrelated
Training-set$_1$	1,026	183	34,302
testing-set$_1$	1,583	310	65,268

Table 8.3: Experiment 1: Training-set$_1$ Information Type Pairs and Semantic Relations

Information-type$_{LHS}$	Information-type$_{RHS}$	Semantic Relation Label
Android ID	Mobile device ID	Direct Hypernymy
Mobile device ID	Device identifier	Direct Hypernymy
DID	Device identifier	Synonymy
URL	URLs	Synonymy
Android ID	Device identifier	Unrelated
Call duration	Advertising ID	Unrelated

number of pairs identified for each class in training-set$_1$ and testing-set$_1$. In addition, Table 8.3 presents examples of information type pairs along with their relationships in training-set$_1$.

In this experiment, we aim to answer the following re-search questions.

RQ1: What is the precision, recall, and F-1 score for the predicted relations?

RQ2: How well the relation classification model can reduce the manual ontology construction effort?

RQ3: What is the effect of missing transitive hypernymy on classification performance?

For experiment 1, we identify the best model configuration based on classification performance (i.e., average F-1 score on three classes) on testing-set$_1$. To this end, we use grid search over six hyper-parameters of relation classification model, including number of epochs, dropout keep rate, batch size, learning rate, convolution activation function, and prediction function. The parameters, their different configuration options, and best performing selections are shown in Table 8.4.

8.2.5 Experiment 2

In experiment 2, we aim to classify whether a new information type pair is one of hypernymy, synonymy, or unrelated. This experiment diverges from experiment 1 by listing both direct and

Table 8.4: Experiment 1: Parameter Configuration Options and Selections

Model Hyper-parameter	Hyper-parameter Options	Best Hyper-parameter Selection
Number of Epochs	10, 15	10
Dropout Keep Rate	0.7, 0.8, 0.9	0.9
Batch Size	30, 128, 200	128
learning Rate	0.01, 0.001	0.001
Convolution Activation Function	tanh, relu, sigmoid	tanh
Prediction Function	sigmoid, softmax	sigmoid

Table 8.5: Experiment 2: Number of Hypernym, Synonym, and Unrelated Pairs in Training and Testing Sets

	Direct Hypernymy	Synonymy	Unrelated
Training-set$_2$	3,827	183	31,501
testing-set$_2$	7,070	310	59,781

transitive hypernymy relationships from a TBox entailment. Therefore, we define hypernymy relationship between two concepts C, D, such that: (1) $C \sqsubseteq D$; and (2) $C \not\equiv D$. We define synonymy relationship for concepts C, D if $C \equiv D$. If a pair is related in any way other than a hypernymy or synonymy relationship, we classify this relationship unrelated. Using this definition, we create training-set$_2$ and testing-set$_2$ using training and testing ontologies (see Table 8.5 for the resulting counts). Additionally, Table 8.6 presents example pairs in training-set$_2$. In contrast to instances listed in Table 8.3, the pair Android ID, device identifier is labeled as hypernymy in experiment 2.

Experiment 2 raises the following research question based on the role of transitive hypernymy

Table 8.6: Experiment 2: Training-set$_2$ Information Type Pairs and Semantic Relations

Information-type$_{LHS}$	Information-type$_{RHS}$	Semantic Relation Label
Android ID	Mobile device ID	Hypernymy
Mobile device ID	Device identifier	Hypernymy
Android ID	Device identifier	Hypernymy
DID	Device identifier	Synonymy
URL	URLs	Synonymy
Call duration	Advertising ID	Unrelated

Table 8.7: Experiment 2: Parameter Configuration Options and Selections

Model Hyper-parameter	Hyper-parameter Options	Best Hyper-parameter Selection
Number of Epochs	10, 15	10
Dropout Keep Rate	0.7, 0.8, 0.9	0.9
Batch Size	30, 128, 200	200
learning Rate	0.01, 0.001	0.001
Convolution Activation Function	tanh, relu, sigmoid	relu
Prediction Function	sigmoid, softmax	softmax

relations:

RQ4: How does entailment in hypernymy affect the performance of relation classification model in terms of precision, recall, and F-1 score?

We also identify the best model configuration based on classification performance (i.e., average F-1 score on three classes) using grid search over six hyper-parameters for experiment 2. The parameters, their different configuration options, and best performing selections are shown in Table 8.7.

Next, we present results for experiments 1 and 2 and address the research questions.

8.3 Experimental Results

In this section, we report our results and answer the research questions described in Sections 8.2.4 and 8.2.5. Recall, we have two experiments: (1) to evaluate hypernymy prediction assuming independent relations; and (2) to evaluate hypernymy prediction assuming dependent relations.

8.3.1 Experiment 1 Results

In experiment 1, we compare the labels of the testing-set$_1$ with the predicted relations to answer **RQ1** and investigate the number of relations correctly predicted by our model. testing-set$_1$ contains 1,583 information type pairs labeled as direct hypernymy, 310 information type pairs labeled as synonymy, and 65,268 pairs as unrelated (see Section 8.2.4 for more details). We use precision, recall, and F-1 score as measures to evaluate the performance of the relation classification model on

Table 8.8: Confusion Matrix for Experiment 1

	Actual Direct Hypernymy	Actual Synonymy	Actual Unrelated	Total Predictions/Class
Predicted Direct Hypernymy	1,228	104	1,207	2,539
Predicted Synonymy	0	0	0	0
Predicted Unrelated	355	206	64,061	64,622
Total Labels/Class	1,583	310	65,268	

Table 8.9: Performance Measures for Experiment 1

	Direct Hypernymy	Synonymy	Unrelated
Precision	0.483	Undefined	0.991
Recall	0.775	0.000	0.981
F-1 Score	0.595	Undefined	0.985

the testing set. Since our dataset is skewed toward unrelated, we opt for F-1 score, which provides a better balance between precision and recall. Table 8.8 presents the confusion matrix, where each row presents the class predictions, and each column presents the actual class instances. For each class, we define correct predictions (CPs), if the prediction is the same as class label, shown in the shaded diagonal. Equations 8.8, 8.9, and 8.10 present performance measure calculations for each relation class (i.e, hypernymy, synonymy, or unrelated), which appear in Table 8.9. The relation classification model fails to predict synonymy relationships. This outcome is not unexpected, since synonyms are rare and the data set is highly imbalanced.

$$Precision_{relation} = \frac{CP_{relation}}{\#Predictions_{relation}} \tag{8.8}$$

$$Recall_{relation} = \frac{CP_{relation}}{\#Labels_{relation}} \tag{8.9}$$

$$F-1_{relation} = 2 \times \frac{Precision_{relation} \times Recall_{relation}}{Precision_{relation} + Recall_{relation}} \tag{8.10}$$

In the manual ontology construction approach [38], an analyst must compare each information type pair to identify a semantic relationship between the types. The cost to compare two information types is reported as 20 seconds on average [37]. Extending an ontology with new information

Table 8.10: Examples of False Hypernymy Predictions

Information-type$_{LHS}$	Information-type$_{RHS}$	Pre	Act	Ent	Hops
Device brand	Device	H	U	Yes	3
Mobile device IP address	Mobile device	H	U	Yes	2
MAC address	Hardware information	H	U	Yes	3
Players interactions	interactions	H	U	Yes	2
Mobile device unique identifier	Device identifiers	H	U	Yes	2
Unique hardware identifiers	Hardware information	H	U	Yes	2
Website activity date	Usage times	H	U	No	-
Devices UDID	Device unique identifier	H	U	No	-
Postal code	Approximate geographical location	H	U	No	-
WiFi signal strength	Mobile device	H	U	No	-
Websites visited	Aggregated user data	H	U	No	-
MAC address	Hardware settings	H	U	No	-

types requires the analyst to compare each new type with all existing types. To address RQ2, we evaluate the extent that the relation classification model can reduce ontology construction effort. The testing ontology contains 367 information types, resulting in $\frac{(367\times(367-1))}{2} = 67,161$ information type pairs that an analyst must evaluate during construction. A key challenge is reducing the number of comparisons, particularly of information type pairs that are unrelated and that dominate the space of comparisons. While the predicted unrelated class includes 30% false positives (561/1,893), it correctly predicts 98% of the unrelated comparisons (64,061/65,268). This trade-off in precision reflects a significant time savings of 355 hours of comparison to correctly identify unrelated pairs.

In experiment 1, the training-set$_1$ and testing-set$_1$ include direct hypernymy relationships and thus label indirect, or ancestor hypernymy relations as unrelated. This task aims to predict the graph edges in an ontology, ignoring that subsumption is transitive and the transitive closure of a concept's class relationships. To answer **RQ3**, we analyzed the logical entailment of 1,207 falsely predicted hypernymy relations that were labeled unrelated. We utilize OWL API HermiT reasoner[2] for this analysis. The results show 44% (541) of falsely predicted hypernymies with unrelated

[2]http://www.hermit-reasoner.com/java.html

Table 8.11: Confusion Matrix for Experiment 2

	Actual Hypernymy	Actual Synonymy	Unrelated	Total Predictions/Class
Predicted Hypernymy	6,098	103	918	7,119
Predicted Synonymy	0	0	0	0
Predicted Unrelated	972	207	58,863	60,042
Total Labels/Class	7,070	310	59,781	

label are logically entailed through indirect hypernymy. Table 8.10 shows a sample analysis result, where the model prediction (Pre) is hypernymy (H) and the actual labeled (Act) is unrelated (U). If a pair is logically entailed, we show the number of hops/edges between the types in the ontology. Notably, predicting indirect hypernyms could improve results.

8.3.2 Experiment 2 Results

In experiment 2, we include direct and indirect hypernymy in the actual labeled relations. The model is trained on training-set$_2$ containing 3,827 information type pairs labeled as hypernymy (direct and transitive), 183 information type pairs labeled as synonymy, and 31,501 pairs as unrelated. We evaluate the performance of the relation classification model by comparing the actual labels of testing-set$_2$ with the predicted relations to address **RQ4**.

We use precision, recall, and F-1 score as measures to evaluate the performance on testing-set$_2$. The confusion matrix for this evaluation is presented in Table 8.11, where each row presents the predictions class counts, and each column presents the actual labeled instances. In contrast to Table 8.11, some portion of unrelated labeled instances are shifted toward hypernymy. Hence, the model's performance on hypernymy shows a significant improvement as shown in Table 8.12 . Further, we notice an increase of 173% within the misclassified hypernymies using this setup. However, we accept this misclassification as a trade-off for a higher F-1 score.

Table 8.12: Performance Measures for Experiment 2

	Direct Hypernymy	Synonymy	Unrelated
Precision	0>856	Undefined	0.980
Recall	0.862	0.000	0.984
F-1 Score	0.858	Undefined	0.981

8.4 Discussion and Future Direction

We now discuss our results, limitations, and improvement required to address some of the challenges in this chapter.

Regulators and app markets require app developers to describe their data practices in privacy policies. Apart from abstraction, stakeholders can use different words to describe the same data concept. Using semantic relations, such as hypernymy and synonymy, one can formalize relationships between concepts to create a shared understanding. The model proposed herein identifies hypernymy relationships between information type pairs with 0.858 F-1 score. The model also identifies unrelated information type pairs with 0.981 F-1 score, greatly reducing the search space of candidate relationships.

We now list the challenges and limitations of our work. Based on the results in experiments 1 and 2, the model failed to predict synonymy relationships due to insufficient training examples. Despite biasing weights for prediction, which enhances hypernymy predictions, synonymy predictions remain unaffected. To reduce the effect of minority class, over-sampling methods, such as random over-sampling and Synthetic Minority Over-Sampling Technique (SMOTE) [18], can be used. Random over-sampling simply repeats training examples to reduce imbalance. In contrast, SMOTE generates synthetic training examples based on distance functions.

In experiment 2, our model misclassifies 972 information type hypernymy pairs as unrelated. One approach to reduce this misprediction could be to use our syntax-based method in Chapters 6 and 7 that infers semantic relations among types that share at least one word [37]. To this end, we utilized the ontology constructed upon lexicon L (see Section 5) to investigate if the misclassified hypernymies can be inferred. Based on the results, we can only infer 56/972 of hypernyms. Alternatively, we may explore the use of knowledge about siblings and shared ancestors to identify pairs misclassified as unrelated. Finally, the manually constructed ontology may include omissions, since manual classification is subject to fatigue and recency effects, where the analysts recalls the information types that they classified last [38, 49]. We propose to add additional forms of evaluation using preference theory [61] to elicit the relationships among predicted classifications. The results of a preference survey can illuminate the extent of relatedness among pairs and evaluate a sample of potential misclassifications.

Finally, the model is trained on platform information types, which does not include domain-specific information types, such as health-, finance-, dating-, and shopping-related app data, to name a few.

Our work has broader impact on data flow and requirements traceability domains. We plan to integrate our work with data flow and requirements compliance detection tools that rely on precise description of information types. For example, Breaux et al. identifies data flow traces across privacy policies of a multi-tier service systems [12]. Further, Slavin et al. and Wang et al. [60, 69] investigate privacy requirements traceability in Android apps.

Chapter 9: FUTURE WORK

In this section, we list two main future directions. The first shows the application of ontologies in identifying data practices with regards to a specific information type. The second direction highlights the application of ontologies in generating privacy snippets.

9.1 Summarizing Privacy Policies

Requirements analysts, regulators, and users might be interested in data practices of an app regarding an information type, such as “network information” or “device information.” To automatically detect the collection, usage, and sharing practices for any information type, a requirements analysis tool can search for privacy statements where the exact match of the information type appears. However, using the information type ontology, the tool can also look for privacy statements that contain specific kinds of that information type. For example, the tool can look for statements containing “network information” as an exact match of the information type, along with the statements containing “IP address” and “MAC address” as specific kinds of “network information.” Using the privacy policy ontology, stakeholders can grasp a more comprehensive view of data practices regrading an information type in apps.

9.2 Generating Controlled Privacy Snippets

Controlled natural languages are designed to improve communication among humans, especially for stakeholders that might use different phrases for the similar domain concept. The restrictions enforced on the language increase the shared understating. We aim to employ this idea to generate privacy policy snippets describing data practices in brief which are easy to comprehend by users. These snippets can also be used toward generating consent that can be presented to users during usage.

To this end, first, we plan to utilize the privacy policy ontology to capture all data practices regarding an information type as mentioned in Section 9.1. With privacy snippets or consent, the challenge is to be comprehensive (i.e., summarizing the data practices into a short text), but addressing the right information types with respect to users' perceived risk on information types and actions. For example, within consent text, it is unreasonable to list all the information types that are either specific kinds or parts of "network information." Thus, we need to identify a balance where the selected information types convey the users perceived risk for the data practice. For example, users might be willing to share their "network information" with third parties, however, they are less willing to share their "MAC address, " although "MAC address" is a specific kind of "network information." More studies on perceived privacy risk have been conducted by Bhatia and breaux [7]. The results of these studies can be used to identify a minimal set of information types for generating transparent privacy snippets without altering or diminishing users' perceived risks toward the practice.

Chapter 10: CONCLUSION

Almost every major privacy law requires companies to disclose their activities surrounding personal data in a transparent way. However, privacy policies as a major source to communicate data practices often fail to provide detailed and transparent description of companies' practices regrading personal data. To be comprehensive, these policies mainly focus on abstract information types, hence reducing shared understanding between different stakeholders, such as requirements analysts, policy authors, app developers, regulators, and users. Specifically on users' side, full and transparent disclosure of data practices allows individuals to make better, informed decisions.

To address the abstraction and variability in concept and achieve a shared understanding of companies' data practices, this thesis proposes methods to capture and formalize the semantics of natural language privacy policies into a knowledge base, called ontology. Arranging terminology into a hierarchical organization (i.e., an ontology) captures relations, such as subclass/superclass, part/whole, and synonymy, among categories. Constructing an empirically-valid ontology is a challenging task since it should be both scalable and consistent with multi-user interpretations. Within this thesis, we discuss different methods to construct privacy ontologies using (1) heuristics derived from grounded analysis of privacy polices, (2) natural language patterns, and (3) neural network classifier. In addition, we also evaluate each proposed method empirically and address the limitations and future directions. Further, we propose a method to split an ontology into training and testing sets for machine learning purposes.

In summary, the proposed work enhances identification of corrective actions that mitigate or eliminate the impacts of privacy risk in software development process using natural language processing and empirical data collection methods. We envision that the results and artifacts produced through this research can be applied in data flow and privacy violation detection tools, enabling companies to align their privacy policies with their actual data practices.

BIBLIOGRAPHY

[1] Annie I Anton and Julia B Earp. A requirements taxonomy for reducing web site privacy vulnerabilities. *Requirements Engineering*, 9(3):169–185, 2004.

[2] Mark Aronoff. Word formation in generative grammar. *Linguistic Inquiry Monographs Cambridge, Mass.*, (1):1–134, 1976.

[3] Franz Baader, Diego Calvanese, Deborah McGuinness, Peter Patel-Schneider, and Daniele Nardi. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.

[4] Emmon Bach. An extension of classical transformational grammar. 1976.

[5] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[6] Jaspreet Bhatia and Travis D Breaux. Towards an information type lexicon for privacy policies. In *Requirements Engineering and Law (RELAW), 2015 IEEE Eighth International Workshop on*, pages 19–24. IEEE, 2015.

[7] Jaspreet Bhatia and Travis D Breaux. Empirical measurement of perceived privacy risk. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 25(6):34, 2018.

[8] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[9] Stephen Boyd, Didar Zowghi, and Vincenzo Gervasi. Optimal-constraint lexicons for requirements specifications. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 203–217. Springer, 2007.

[10] Travis D Breaux, Annie I Antón, and Eugene H Spafford. A distributed requirements management framework for legal compliance and accountability. *computers & security*, 28(1-2):8–17, 2009.

[11] Travis D Breaux and David L Baumer. Legally "reasonable" security requirements: A 10-year ftc retrospective. *Computers & Security*, 30(4):178–193, 2011.

[12] Travis D Breaux, Hanan Hibshi, and Ashwini Rao. Eddy, a formal language for specifying and analyzing data flow specifications for conflicting privacy requirements. *Requirements Engineering*, 19(3):281–307, 2014.

[13] Travis D Breaux and Florian Schaub. Scaling requirements extraction to the crowd: Experiments with privacy policies. In *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, pages 163–172. IEEE, 2014.

[14] Travis D Breaux and Florian Schaub. Scaling requirements extraction to the crowd: Experiments with privacy policies. In *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, pages 163–172. IEEE, 2014.

[15] Travis D Breaux, Daniel Smullen, and Hanan Hibshi. Detecting repurposing and over-collection in multi-party privacy requirements specifications. In *2015 IEEE 23rd international requirements engineering conference (RE)*, pages 166–175. IEEE, 2015.

[16] Karin K Breitman and Julio Cesar Sampaio do Prado Leite. Ontology as a requirements engineering product. In *Proceedings. 11th IEEE International Requirements Engineering Conference, 2003.*, pages 309–319. IEEE, 2003.

[17] Razvan C Bunescu and Raymond J Mooney. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 724–731. Association for Computational Linguistics, 2005.

[18] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[19] Chunyang Chen, Zhenchang Xing, and Ximing Wang. Unsupervised software-specific morphological forms inference from informal discussions. In *Proceedings of the 39th International Conference on Software Engineering*, pages 450–461. IEEE Press, 2017.

[20] Noam Chomsky. *Aspects of the Theory of Syntax*, volume 11. MIT press, 2014.

[21] Juliet Corbin, Anselm Strauss, and Anselm L Strauss. *Basics of qualitative research*. Sage, 2014.

[22] Ferdinand De Saussure and Roy Harris. Course in general linguistics.(open court classics). *Chicago and La Salle, Open Court*, 1998.

[23] Morgan C Evans, Jaspreet Bhatia, Sudarshan Wadkar, and Travis D Breaux. An evaluation of constituency-based hyponymy extraction from privacy policies. In *Requirements Engineering Conference (RE), 2017 IEEE 25th International*, pages 312–321. IEEE, 2017.

[24] Dieter Fensel. Ontologies: A silver bullet for knowledge management and electronic-commerce (2000). *Berlin: Spring-Verlag*, 143.

[25] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[26] Gottlob Frege. Über begriff und gegenstand. 1892.

[27] Peter Gerstl and Simone Pribbenow. A conceptual theory of part-whole relations and its applications. *Data & Knowledge Engineering*, 20(3):305–322, 1996.

[28] Vincenzo Gervasi and Didar Zowghi. On the role of ambiguity in re. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 248–254. Springer, 2010.

[29] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[30] Jin Guo, Jinghui Cheng, and Jane Cleland-Huang. Semantically enhanced software traceability using deep learning techniques. In *Proceedings of the 39th International Conference on Software Engineering*, pages 3–14. IEEE Press, 2017.

[31] Zhou GuoDong, Su Jian, Zhang Jie, and Zhang Min. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 427–434. Association for Computational Linguistics, 2005.

[32] Hamza Harkous, Kassem Fawaz, Rémi Lebret, Florian Schaub, Kang G Shin, and Karl Aberer. Polisis: Automated analysis and presentation of privacy policies using deep learning. *arXiv preprint arXiv:1802.02561*, 2018.

[33] Kamala D Harris. *Privacy on the go: recommendations for the mobile ecosystem*. 2013.

[34] Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics, 1992.

[35] Barendregt Henk. The lambda calculus: its syntax and semantics. *Studies in logic and the foundations of Mathematics*, 1984.

[36] Christopher Hookway. *Peirce-Arg Philosophers*. Routledge, 2010.

[37] Mitra Bokaei Hosseini, Travis D Breaux, and Jianwei Niu. Inferring ontology fragments from semantic role typing of lexical variants. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 39–56. Springer, 2018.

[38] Mitra Bokaei Hosseini, Sudarshan Wadkar, Travis D Breaux, and Jianwei Niu. Lexical similarity of information type hypernyms, meronyms and synonyms in privacy policies. In *2016 AAAI Fall Symposium Series*, 2016.

[39] Chu-Ren Huang. *Ontology and the lexicon: a natural language processing perspective*. Cambridge University Press, 2010.

[40] Theo MV Janssen and Barbara H Partee. Compositionality. In *Handbook of logic and language*, pages 417–473. Elsevier, 1997.

[41] Dan Jurafsky and James H Martin. *Speech and language processing*, volume 3. Pearson London, 2014.

[42] Yann LeCun et al. Generalization and network design strategies. *Connectionism in perspective*, pages 143–155, 1989.

[43] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636. ACM, 2006.

[44] Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304, 2015.

[45] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[46] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[47] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[48] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual*

Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2, pages 1003–1011. Association for Computational Linguistics, 2009.

[49] Bennet B Murdock Jr. The serial position effect of free recall. *Journal of experimental psychology*, 64(5):482, 1962.

[50] Preslav Nakov and Marti Hearst. Using verbs to characterize noun-noun relations. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, pages 233–244. Springer, 2006.

[51] Frank Olken and Doron Rotem. Random sampling from databases: a survey. *Statistics and Computing*, 5(1):25–42, 1995.

[52] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[53] Gabriele Petronella. Analyzing privacy of android applications. 2014.

[54] Leo Postman and Laura W Phillips. Short-term temporal changes in free recall. *Quarterly journal of experimental psychology*, 17(2):132–138, 1965.

[55] Simone Pribbenow. What's a part? on formalizing part-whole relations. In *Foundations of Computer Science*, pages 399–406. Springer, 1997.

[56] Lee Rainie and Andrew Perrin. facts about smartphones as the iphone turns 10. *Pew Research Center*, 10.

[57] Siegfried Rasthofer, Steven Arzt, and Eric Bodden. A machine-learning approach for classifying and categorizing android sources and sinks. In *2014 Network and Distributed System Security Symposium (NDSS)*, 2014.

[58] Subhro Roy and Dan Roth. Mapping to declarative knowledge for word problem solving. *Transactions of the Association of Computational Linguistics*, 6:159–172, 2018.

[59] Johnny Saldaña. *The coding manual for qualitative researchers*. Sage, 2015.

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

[61] Paul Slovic. The construction of preference. *American psychologist*, 50(5):364, 1995.

[62] Rion Snow, Daniel Jurafsky, and Andrew Y Ng. Learning syntactic patterns for automatic hypernym discovery. In *Advances in neural information processing systems*, pages 1297–1304, 2005.

[63] Péter Szeredi, Gergely Lukácsy, and Tamás Benkő. *The Semantic Web explained: the technology and mathematics behind Web 3.0.* Cambridge University Press, 2014.

[64] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[65] Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432, 2015.

[66] Duyu Tang, Bing Qin, and Ting Liu. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1014–1023, 2015.

[67] Mike Uschold and Michael Gruninger. Ontologies: Principles, methods and applications. *The knowledge engineering review*, 11(2):93–136, 1996.

[68] Nicolas Viennot, Edward Garcia, and Jason Nieh. A measurement study of google play. In *ACM SIGMETRICS Performance Evaluation Review*, volume 42, pages 221–233. ACM, 2014.

[69] Xiaoyin Wang, Xue Qin, Mitra Bokaei Hosseini, Rocky Slavin, Travis D Breaux, and Jianwei Niu. Guileak: Tracing privacy policy claims on user input data for android applications. In *Proceedings of the 40th International Conference on Software Engineering*, pages 37–47. ACM, 2018.

[70] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *Journal of machine learning research*, 3(Feb):1083–1106, 2003.

[71] Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. Relation classification via convolutional deep neural network. In *COLING*, pages 2335–2344, 2014.

[72] Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212, 2016.

[73] Sebastian Zimmeck, Ziqi Wang, Lieyong Zou, Roger Iyengar, Bin Liu, Florian Schaub, Shomir Wilson, Norman Sadeh, Steven Bellovin, and Joel Reidenberg. Automated analysis of privacy requirements for mobile apps. In *2016 AAAI Fall Symposium Series*, 2016.

[74] Sebastian Zimmeck, Ziqi Wang, Lieyong Zou, Roger Iyengar, Bin Liu, Florian Schaub, Shomir Wilson, Norman Sadeh, Steven M. Bellovin, and Joel Reidenberg. Automated analysis of privacy requirements for mobile apps. In *Network and Distributed System Security Symposium NDSS*, 2017.

VITA

Mitra Bokaei Hosseini was born in Tehran, Iran. She graduated from the Azad University of Qazvin in Iran in 2008 with a B.S. in Information Technology. Starting her Master's degree in Information Technology with e-commerce concentration, she graduated from K. N. Toosi University of Technology in Tehran in June 2011. Mitra has experience working in industry as Information Security auditor for four years. She joined the Computer Science Ph.D. program at University of Texas at San Antonio in Spring 2014. Her research spans on requirements engineering, privacy engineering, and natural language processing. Mitra has worked as a research intern with research institutes and universities in the U.S. She looks forward to a future in academia both as a researcher and teacher.