Ambiguity and Generality in Natural Language Privacy Policies

Mitra Bokaei Hosseini
Computer Science Department
St. Mary's University
San Antonio, TX, United States
mbokaeihossein@stmarytx.edu

John Heaps, Rocky Slavin, Jianwei Niu
Computer Science Department
University of Texas at San Antonio
San Antonio, TX, United States
{john.heaps,rocky.slavin,jianwei.niu}@utsa.edu

Travis Breaux
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, United States
breaux@cs.cmu.edu

Abstract—**Privacy policies are legal documents containing application data practices. These documents are well-established sources of requirements in software engineering. However, privacy policies are written in natural language, thus subject to ambiguity and abstraction. Eliciting requirements from privacy policies is a challenging task as these ambiguities can result in more than one interpretation of a given information type (e.g., ambiguous information type "device information" in the statement "we collect your device information"). To address this challenge, we propose an automated approach to infer semantic relations among information types and construct an ontology to guide requirements authors in the selection of the most appropriate information type terms. Our solution utilizes word embeddings and Convolutional Neural Networks (CNN) to classify information type pairs as either hypernymy, synonymy, or unknown. We evaluate our model on a manually-built ontology, yielding predictions that identify hypernymy relations in information type pairs with 0.904 F-1 score, suggesting a large reduction in effort required for ontology construction.**

Index Terms—**Privacy Policy; Privacy Requirement; Ambiguity; Generality; Semantic Relation; Neural Network; Ontology**

I. Introduction

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

The challenge of acquiring requirements from data practice descriptions, however, is that privacy policies often contain ambiguities [5], which admit more than one interpretation [6]. Furthermore, policies are intended to generalize across a wide range of data practices, and are not limited to describe a single software system, in which case they also exhibit vagueness and generality [7]. Berry and Kamsties distinguish four broad categories of linguistic ambiguity, including lexical, syntactic, semantic, and pragmatic ambiguity [8]. They further separate vagueness and generality from ambiguity. *Vagueness* occurs when a phrase admits borderline cases, e.g., the word "tall" is vague when considering a subject who is neither tall nor not tall [8]. In *generality*, a superordinate term refers to two or more subordinate terms. In linguistics, generality is encoded by the relationship between a *hypernym*, or the general term, and more specific terms, called *hyponyms*.

In privacy policies, information types can be expressed using both vague and general terms. Many policies describe the vague phrase "device information," which can potentially include both a device's "location" and its "IMEI number," which users may consider more or less private, leading to boundary cases. In addition, they contain general terms, such as "address," which are intended to refer to more specific meanings, such as "postal address," "e-mail address," or "network address," in which case the reader must choose an interpretation to fit the given context.

Ambiguity, generality, and vagueness have been extensively studied in requirements engineering research, particularly in regulatory and policy documents. This includes techniques to identify, classify, and model ambiguity in regulations, such as HIPAA [9], [5], and techniques to identify generality [3], [10], [11] and vagueness [12] in privacy policies. Recently, two studies employ hand-crafted regular expressions over nominals, and constituency parse trees derived from individual policy statements to extract generalities, specifically hypernyms[13], [14]. In another study, Hosseini et al. employ a context-free grammar and semantic attachments to infer generalities between information types that share at least one common word (e.g., "device identifier" and "identifier") [15]. This study solely relies on syntax information to infer hypernyms. Consequently, generality relations between information types, such as "device identifier" and "MAC address," that rely on contextual semantics and tacit knowledge are ignored.

To address the problem of ambiguity in privacy policy terminology, we propose a context-wise semantic model to identify semantic relationships, such as hypernymy and synonymy, between *information types in a given pair* (e.g., pair $\langle$mobile device identifier, MAC address$\rangle$). Such relationships are then formalized into partial ontologies which can be used by requirements engineers for terminology disambiguation. This model consists of five layers: information type pair input layer, embedding layer, phrase modeling layer (i.e., convolutional neural network (CNN)), semantic similarity and classification layer, and finally an output layer that specifies the relation as hypernymy, synonymy, or unknown. We investigate the effectiveness of this model using a *manually-constructed ontology* as our ground truth. We elicit all the information type pairs and their relations from this ontology, which are then used as training and testing sets in two experiments. Our experiments show that this model significantly outperforms the syntax-driven method proposed by Hosseini et al. [15] with 86% precision and 95% recall for hypernymy. The results confirm that capturing information type contextual representation from privacy policies can facilitate the ontology construction task.

The main contributions of this paper are as follows: (1) Contextual representation of information types using Word2Vec and CNN for relationship classification; (2) Context-wise model to classify relations given an information type pair; (3) An empirical evaluation of the context-wise model on the manually-constructed ontology.

This paper is organized as follows. In §II, we motivate using ontologies in RE. In §III and §IV, we discuss terminology and related work. In §V, we introduce our model. In §VI and §VII, we present the experimental design and results, followed by threats, discussion, and conclusion in §VIII, §IX, and §X.

II. Ontology Usage Motivation

Ambiguity in privacy policies creates challenges for managing privacy, tracing privacy requirements to data practices in code, and checking the compliance of privacy policies with regulations [9], [16], [17]. Policy analysis tools such as `PVDetector` identify misalignments between policy and practice [18]. This tool maps information types in privacy policies to privacy-related API methods implemented in the corresponding code. This mapping provides the semantics needed to check code for misalignment with privacy policy, and to suggest where the code or policy may be changed to fit the functional and legal requirements of apps [18]. The tool utilizes an ontology as a *resource* to identify relations between information types in privacy policies.

Given the app's byte code (i.e., apk file) and corresponding privacy policy, `PVDetector` generates a list of detected misalignments. These misalignments are marked as either *strong* (i.e., no information type related to an API method invocation is found in the privacy policy) or a *weak* (i.e., no information type *directly mapped* to an API method invocation is found, but some more general information type is found in the privacy policy). For example, assume an app has the following text in its privacy policy. "*We collect browsing history, contact information, **mobile identifiers**, and language information.*" If, upon analysis of the app's actual dataflows, a flow is detected from the method `getMacAddress()` to some method that sends data away through the network, the possibility of private information leakage implied. Therefore, some corresponding information type (e.g., "MAC address") should appear in the privacy policy as information that may be collected. Instead, the more broad information type "mobile identifiers", which does not *directly* describe "MAC address" but is *generally* representative of it, appears in the policy. Without considering the hypernymy relation between "MAC address" and "mobile identifiers" (i.e., MAC address is a *kind of* mobile identifier), an incorrect misalignment of omission by the policy would be detected (i.e., false positive). An ontology containing such relationships facilitates the detection of the hypernym in the policy. Thus, the false negative due to the use of a more general term in the policy is avoided, and the policy writer can be made aware of the more specific term "MAC address" for improved clarity and precision. In summary, to support automated analysis of requirements, tools and techniques are needed to build and validate formal ontologies.

III. Background

Ambiguity: Berry and Kamsties distinguish four broad categories of linguistic ambiguity, including lexical, syntactic, semantic, and pragmatic ambiguity [8]. They also discuss two phenomena closely related to ambiguity, including vagueness and generalization [8], [6]. A requirement is ambiguous if it admits more than one interpretation [19]. A requirement is unambiguous if different stakeholders with similar backgrounds give the same interpretation to it [20].

Semantic relations: (1) *Hypernymy*- a relationship between two noun phrases where the meaning of one (hypernym) is more generic than the other (hyponym), e.g., "device information" is a hypernym of "device ID." (2) *Synonymy*- a relationship between two noun phrases with a similar meaning or an abbreviation, e.g., "IP" and "Internet protocol."

Lexicon: a collection or list of information type phrases.

Ontology: an arrangement of concept names in a graph in which terms are connected via edges corresponding to relationships, such as hypernymy [21]. In this paper, we only consider information type names as concept names.

Word Embedding: Distributed representation of a word as a vector in some m-dimensional space that helps learning algorithms achieve better performance by grouping similar words together [22], [23]. Each vector dimension represents some feature of the words' semantics in a corpus. Skip-gram is a popular word embedding model [24]. For each word in a corpus, the surrounding words (identified by a window size) are used as *context* for that word. This context is then used as input to a neural network that will modify the word's vector values. We adopt Word2Vec[1], an implementation of the Skip-gram model to construct domain-specific word embeddings.

Convolutional Neural Network (CNN): CNNs are a kind of feed-forward network, specialized in processing data with a grid-like topology [25]. 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 [26]. The second step applies a non-linear function (e.g., tanh, relu, etc.) to each linear activation produced by the previous step. In the third step, different types of pooling functions (e.g., max pooling, average pooling, etc.) are applied to sets of areas, 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 [26]. The matrix result after these three steps is a representation of the main features of the input.

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

IV. Related Work

A. Detection and Resolution of Ambiguity

Detection and resolution of ambiguity in requirements has been the subject of studies in the RE community for decades. Lami et al. [27] automatically detect potential linguistic defects that can determine ambiguity. Nigam et al. identify lexical, syntactic, and syntax ambiguities in words and provide the possible sources of wrong interpretation in requirements using POS tagger and a corpus of ambiguous words [28]. Kiyavitskaya et al. [29] measure lexical and syntactic ambiguities defined by [8] and identify specific instances of pragmatic, software-engineering, and language-error ambiguities in sentences. Ferrari and Gnesi [30] detect pragmatic ambiguities using n knowledge graphs built upon n individual requirement documents. The knowledge graphs are not designed to formalize the semantic relations between terms. Ferrari et al. [31] identify and categorize ambiguity in requirements elicitation interviews. Massey et al. [5] model legal text using ambiguity taxonomy introduced in [9]. Boyd et al. reduce ambiguity in controlled natural languages by optimally constraining lexicons using term *replaceability* [32]. Bhatia et al. [12] introduce a theory of vagueness for privacy policies using a taxonomy of vague terms derived manually.

Historically, WordNet [33], [34] is widely used in detecting ambiguity [29] and is also reported as the most utilized lexicon to support NLP-related RE tasks [35]. However, further analysis reveals that only 14% of phrases from a privacy policy lexicon are found in WordNet [10]. Evans et al. apply patterns to privacy policies to extract hypernymy pairs [14]. Pattern sets are limited because they must be manually extended to address new policies. Hosseini et al. [13], [15] propose regular expression patterns and a context-free grammar (CFG) to parse a given information type and infer semantic relations based on syntax to construct partial ontologoies. The patterns and CFG rules fail to infer semantic relations beyond syntax that require tacit knowledge. Besides, the methods requires a manual tagging step, where each word is tagged with a syntactic role. Our proposed model overcomes these shortcomings and provides a fully automated approach to infer hypernymy relations.

B. Ontology in Requirements Modeling and Analysis

Ontologies are a standard form for representing the concepts within a domain, as well as the relationships between those concepts in a way that allows automated reasoning [36]. Due to such benefits, prior work in RE has employed ontology in requirements formalization and modeling. For example, Gordon and Breaux [37] formalize regulatory requirements from multiple jurisdictions into a single standard of care. Breitman and do Prado Leite describe how ontologies can be used to analyze web application requirements [38]. Breaux et al. use an ontology to identify conflicting requirements across vendors in a multi-stakeholder data supply chain [3]. Oltramari et al. propose a formal ontology to specify privacy-related data practices and their categories [39]. However, this ontology does not entail semantic relations among information types. Humphreys et al. [40] semi-automatically populate legal ontologies by extracting definitions, norms, and other elements of regulations using semantic role labeling.

Ontologies have been applied in tracing requirements to data practices expressed in source code. Two recent works [11], [41] identify app code that is inconsistent with privacy policies using manually-constructed ontologies [10]. These works exemplify the efficacy of ontologies for requirements traceability. However, the manual construction of ontologies is costly and lacks scalability due to the time spent by analysts to compare information types, and errors generated by analysts during comparison [10]. Our proposed automated model is an improvement on prior work to construct ontologies.

C. Relationship Extraction using Machine Learning (ML)

Snow et al. employ hypernym-hyponym pairs in WordNet to identify additional pairs in the parsed sentences of the Newswire corpus [42]. The approach relies on the explicit expression of hypernymy pairs in text. Hearst propose six lexico-syntactic patterns to automatically identify hypernymy in text using noun phrases and regular expressions [43]. Traditional ML approaches classify the relationship between a pair of words in sentences by extracting features, such as part-of-speech tags, shortest dependency path, and named entities [44], [45], [46], [47]. Some models use deep learning to learn sentence-level semantics of word pairs [48], [49]. Recent works are sentence-independent and employ general-purpose distributional vectors for a pair of words as features [50], [51], [52], [53]. The vectors are learnt from Wikipedia, WordNet, or Newswire corpus. In contrast, our proposed model is designed on domain specific phrases, such as "MAC address" and learns the vectors from a privacy policy corpus.

V. Context-wise Relation Classification Model

We propose a novel context-wise model for inferring semantic relations between two given information types as a pair. Figure 1 shows the overview for the model with a pair of information types as input. Throughout the paper, we present the information type pair as $\langle$information-type$_{LHS}$, information-type$_{RHS}\rangle$, e.g., $\langle$device information, device ID$\rangle$, 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 unknown. We select the most probable relation for each information type pair. We now describe each step in detail.

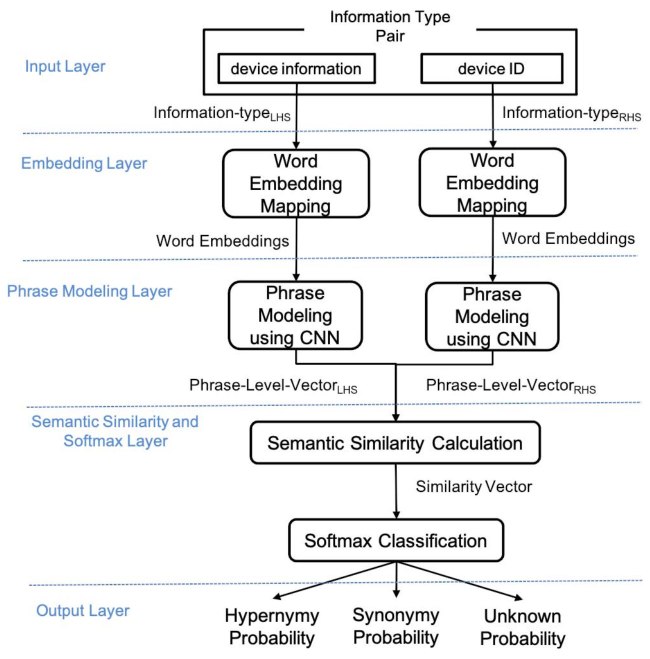

Fig. 1. Context-wise Relation Classification Model

A. Word Embedding Mapping

Each word in an information type phrase is presented using a pre-trained 200-dimensional vector called a word embedding [23]. To create domain-specific word embeddings, we train the Word2Vec model using 77,556 English privacy policies collected from mobile applications on the Google Play Store [2] [54]. Common-purpose word embeddings trained on the English Wikipedia dump [55], [56], [57] or Google News dataset [22] exist, however, previous research has shown improvements on classification accuracy by utilizing domain-specific word embeddings [58].

To obtain the privacy policy corpus to train the Word2Vec model, we crawl the metadata archive for more than 1,402,894 Android apps provided by the PlayDrone project [59] from which 109,933 contained a valid link to a privacy policy. We use the BeautifulSoup library in Python to 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 tokenize the sentences and remove all non-English sentences. We also expand the contractions (e.g., “won’t” is transformed to “will not”), and remove punctuation, numbers, email addresses, URLs, and special characters. Finally, we transform the remaining characters to lower-case. The resulting pre-processed text is used to train the Word2Vec [22] 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.

[2]http://play.google.com

B. Phrase Modeling using CNN

In this section, we describe the phrase modeling architecture (see Figure 2) that transforms word embeddings of an input information type to a low dimensional, fixed-sized vector using Convolutional Neural Network (CNN) with three different filter widths [58]. Implementing CNN with multiple filter widths captures local semantics of n-gram of various granularities [60]. In our case, convolutional filters with widths 1, 2, and 3 capture the semantics of unigrams, bigrams, and trigram, respectively.

We present an example for convolution filter of width $w = 3$ for the padded information type P: device$_1$ information$_2$ pad$_{t-1}$ pad$_t$ with length t, where t is the maximum phrase length in the privacy policy lexicon as discussed in Section V-A. The words/pads in the information type P are represented as a list of vectors $(x_1, x_2, ..., x_{t-1}, x_t)$, where $x_i \in \mathbb{R}^n$ corresponds to word embedding of word/pad $i \in P$ and n represents the dimension of word embeddings ($n = 200$ as mentioned in Section V-A). 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 presented as follows: $\{[x_1; x_2; x_3], ..., [x_{t-2}; x_{t-1}; x_t]\}$, where “;” shows vertical vector concatenations. In general, the result of this module is matrix $X \in \mathbb{R}^{n_0 \times t}$, where $n_0 = w \times n$. To convolve all the features in X, we process X using the linear transformation in (1).

$$Z = W_1 X \tag{1}$$

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 model. The result of linear transformation is shown as $Z \in \mathbb{R}^{n_1 \times t}$, which is dependent on t, the maximum phrase length in the privacy policy lexicon.

We further apply $tanh$ as a non-linear activation function, see (2), on the result of the linear transformation.

$$h = tanh(Z) \tag{2}$$

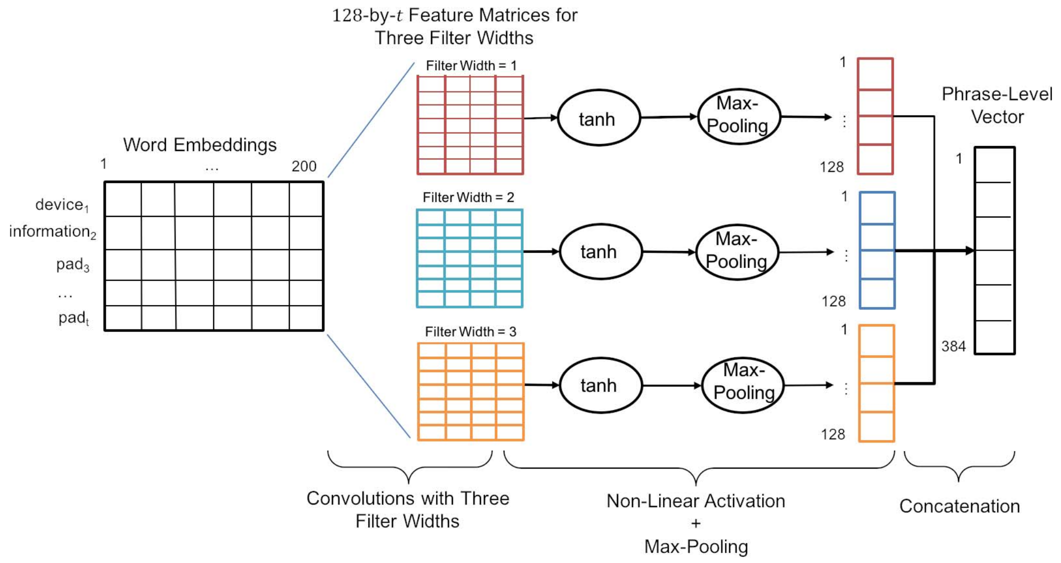

Fig. 2. Phrase Modeling using CNN

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.

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 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.

C. Semantic Similarity Calculation

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. [61], where the direction and distance of the two input vectors are compared using the following equations. Two vectors PLV_{LHS} and PLV_{RHS} refer to Phrase-Level-Vector$_{LHS}$ and Phrase-Level-Vector$_{RHS}$ in Figure 1, which are shortened for simplicity:

$$dir = PLV_{LHS} \odot PLV_{RHS} \quad (3)$$

$$Dis = |PLV_{LHS} - PLV_{RHS}| \quad (4)$$

$$sim = \sigma(Wdir + Udir + b) \quad (5)$$

Equation (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 (4). To integrate the results of (3) and (4) on PLV_{LHS} and PLV_{RHS}, we use a hidden sigmoid layer presented in (5). The similarity vector as the output of the function is then sent to a Softmax classifier as shown in (6) to predict the probabilities of hypernymy, synonymy, and unknown. 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.

$$P_{relation} = softmax(W_p sim + b_p) \quad (6)$$

D. 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, unknown). 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 unknown 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: $unknown/total$ and $(hypernymy + synonymy)/total$. The unknown ratio is applied to 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 unknown class to give less weight when determining the loss. The loss function is defined in (7).

$$loss = -w^i \sum_{i \in T} y^i ln(P^i_{relation}) \quad (7)$$

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 §VI-B and §VI-C. The training process stops when the loss value is sufficiently small or fails to decrease [62].

VI. Experimental Designs

Predicting the ontological relationship between two entities is a multi-class classification problem: is the first item in an ordered information type pair, a hyponym, synonym, or unknown to the second item? Traditional machine learning classification approaches make an independent and identically distributed (IID) assumption for data instances, i.e., predicting the class label of each instance in isolation [63], [64]. In requirements engineering, a major focus is to solve classification problems with IID assumptions. For example, classifying app reviews, bug reports, and feature requests [65], or customer praise [66]; mining Twitter feeds for identifying software requirements [67], [68], [69], [70]; classifying functional and non-functional requirements [71], [72], [73]; and software feature request detection [73].

In contrast, the IID assumption fails in classification problems where class labels are related to features of other labeled instances. These problems can be best described as a set of objects interconnected via links to form a network structure [74]. For example, in hypertext classification, predicting the topic of a webpage requires the knowledge of webpages linked to the page [75]. In a hyperlink network, the classification goal is to label nodes (webpages) with appropriate topics. To achieve this goal, proposed classifiers not only use a webpage's own words, but also consider the neighboring webpages, their attributes, and topics [75], [76], [77]. We extend this view to the relationship classification problem: for each concept in the ontology, we utilize the direct subsumption and equivalence relations, along with the indirect relations with ancestors through transitivity and equalities.

In learning ontologies, we evaluate both views with and without the IID assumption, respectively: (1) each relationship is independent, and the model learns direct relationships among concepts ignoring the transitive closure of hypernymy; or (2) relationships can be dependent, and the model learns a partial semantic representation that is some subset of the transitive closure of hypernymy. For example, we assume the concept pairs $\langle$Android ID, mobile device ID$\rangle$ and $\langle$mobile device ID, device identifier$\rangle$ are related by a hypernymy relationship. We assume the model learns semantic relationships among these words based on how they are used in policy sentences. Under (1) above, we train the model to learn these relationships from policy embeddings. Under (2) above, however, we further train the model to learn relationships inferred through transitivity, which includes the hypernymy relationship for the pair $\langle$Android ID, device identifier$\rangle$. We hypothesize that this additional training generalizes to improve the classification of hypernymy, because more abstract hypernyms can be used to group semantically similar, indirectly related concepts. Furthermore, this approach aligns with our intuition to discover statistical relationships between how abstract terms are used in sentences, and how their concrete examples are used.

Info Type	Frequency
IP address	41
Browser type	21
IP addresses	21
Internet protocol	16
Location information	16

TABLE I: Five Most Frequent Information Types in L

Based on these two views, we design two experiments to evaluate the context-wise relation classification model described in Section V. In experiment 1, we evaluate the model's ability to classify whether an information type pair is a direct hypernymy, synonymy, or unknown. 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 unknown.

This section is organized as follows. We first introduce our ground-truth ontology followed by the approach to design experiment 1. Next, we discuss experiment 2, which unfolds the unique challenge of learning with ontologies and the overall design for experiment 2.

A. Ground-truth Ontology

As our ground truth, we utilize an ontology manually built upon a privacy policy lexicon, called L. The lexicon is extracted from the data collection practices of 50 mobile app privacy policies and contains 356 platform-related information types (e.g., "IP address") defined as "any information that the app or another party accesses through the mobile platform that is not unique to the app" [11]. These 50 policies are unrestricted by domain, and cover gaming, finance, communication, music, productivity, social and entertainment, sports, and shopping, among others. In Table I, we present the top 5 most frequent information types across lexicon L. Hosseini et al. construct an ontology from lexicon L by manually applying seven heuristics that are identified through grounded analysis of five privacy policies [10], [11]. This ontology[3] contains 367 information types, which are used to comprise 1,583 hypernymy and 310 synonymy relationships between information type pairs [10].

We use the following formal representation for an ontology. An ontology is a knowledge base KB expressed using $\mathcal{FL}_0$, a sub-language of the Attribute Language ($\mathcal{AL}$) in Description Logic (DL). A DL knowledge base KB is comprised of two components, $TBox$ and $ABox$ [78]. $TBox$ consists of terminology, i.e., the vocabulary (concepts and

[3]http://polidroid.org/downloads/ontology.owl

	Hypernymy	Synonymy	unknown
$truth\text{-}set_1$	1,583	310	65,268

TABLE II: Experiment 1: Number of Pairs in $truth\text{-}set_1$

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	unknown
Call duration	Advertising ID	unknown

TABLE III: Experiment 1: $truth\text{-}set_1$ Pairs Examples

roles) of an application domain. $ABox$ contains assertions about named individuals using this vocabulary. The manually-constructed ontology only contains terminology, which we call $TBox$ $\mathcal{T}$ [10]. $TBox$ $\mathcal{T}$ contains *terminological axioms* that relate concepts to each other in the form of subsumption and equivalence, which we use to formalize hypernymy, and synonymy. A concept C is subsumed by a concept D, written $\mathcal{T} \models C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$ that satisfy $TBox$ $\mathcal{T}$. The concept C is equivalent to a concept D, written $\mathcal{T} \models C \equiv D$, if $C^{\mathcal{I}} \equiv D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$ that satisfy $TBox$ $\mathcal{T}$. Axioms of the first kind ($C \sqsubseteq D$) are called inclusions, whereas axioms of the second kind ($C \equiv D$) are called equalities [78]. Note that the equalities $C \equiv D$ can be rewritten as two inclusion axioms $C \sqsubseteq D$ and $D \sqsubseteq C$. A subset of the ground-truth ontology is shown in Figure 3. For clarity, we focus on “identifiers” in this figure, hiding other subclasses. Furthermore, some concepts, such as “mobile device identifier” and “mobile device identifiers” are equivalent and this equality is shown using two inclusions.

B. Experiment 1

In this experiment, we evaluate the IID assumption by classifying whether a new information type pair describes a direct hypernymy, synonymy, or unknown. 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 a way other than a direct hypernymy or synonymy relationship, we classify this relationship as unknown. *unknown* means an explicit relationship is yet unknown, yielding no change to the ontology. Using this definition, we identify direct hypernyms, synonyms, and unknown pairs in the ground-truth ontology as $truth\text{-}set_1$. Table II presents the number of pairs identified for each class in $truth\text{-}set_1$. Table III presents examples of information type pairs along with their relationships in this set.

We train and test the classifier with a 10-fold cross validation on $truth\text{-}set_1$. To this end, we split $truth\text{-}set_1$ into two parts with 70:30 ratio. One dataset (70% of $truth\text{-}set_1$) is used for hyper-parameter optimization and training, called $training\text{-}set_1$. The second dataset, called $testing\text{-}set_1$, is set aside toward final testing[4]. For hyper-parameter optimization, we partition $training\text{-}set_1$ into 10 sets $D_1, D_2, D_3, \cdots, D_{10}$. For each possible hyper-parameter combination we use nine sets to fit our model and use the remaining set to validate the model. This process is repeated 10 times for each possible hyper-parameter combination, yielding 10 average F-1 score on three classes. To create possible combinations of hyper-parameters, we use grid search over six hyper-parameters of the classification model, including number of epochs, dropout keep rate, batch size, learning rate, convolution activation function, and prediction function. The hyper-parameters, their different configuration options, and best performing combination with respect to average F-1 score on three classes, are shown in Table IV.

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 IV: Experiment 1: Parameter Configuration Options

After selecting the best hyper-parameter combination, we test the model using $testing\text{-}set_1$ and report the performance metrics for three classes. In this experiment, we aim to answer the following research questions.

RQ1: To what extent does the classification model reduce the manual ontology construction effort?

RQ2: What is the effect of missing transitive hypernymy on classification performance?

C. Experiment 2

In an ontology, the relationships between concepts are not independent, e.g., the relationships in hypernymy are transitive. Removing a relationship between a superordinate and subordinate concept can lead to misclassification between the subordinate concept and its ancestor concepts, because missing intermediary concepts provide the categorical bridge to ancestor concepts. Thus, we are interested in classifying information type pairs using the pair’s attributes, and the pair’s neighbors’ attributes. This approach assumes the IID assumption is false.

Experiment 2 classifies whether a new information type pair is one of hypernymy, synonymy, or unknown. This experiment diverges from experiment 1 by listing both direct and 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

[4]https://github.com/01458198/RE21-Dataset

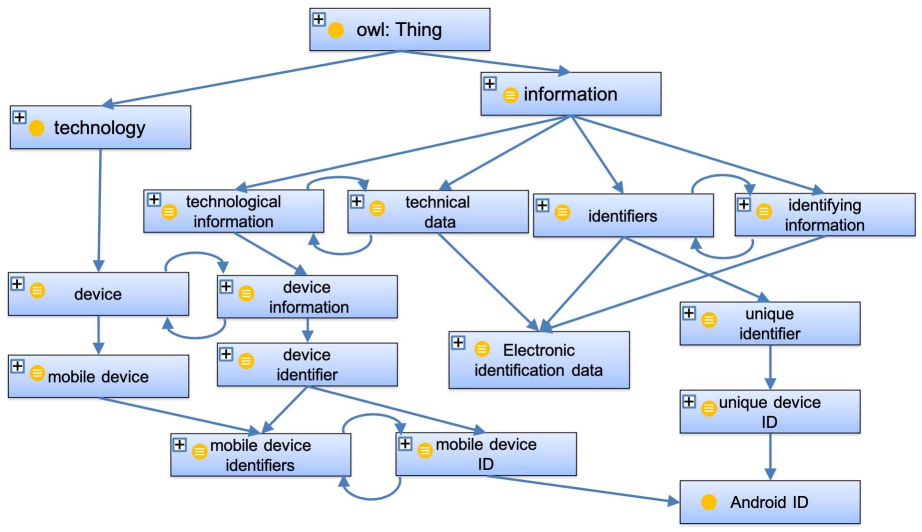

Fig. 3. Ground-truth Ontology Fragment View

	Hypernymy	**Synonymy**	**Unknown**
$truth\text{-}set_2$	7,070	310	59,781

TABLE V: Experiment 2: Number of Pairs in $truth\text{-}set_2$

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	Unknown

TABLE VI: Experiment 2: $truth\text{-}set_2$ Pairs Examples

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 unknown. Using this definition, we create $truth\text{-}set_2$ and present the resulting counts for each relationship in Table V. Further, Table VI presents examples from this set. In contrast to instances listed in Table VI, the pair ⟨Android ID, device identifier⟩ is labeled as hypernymy.

We train and test the classifier with a 10-fold cross validation on $truth\text{-}set_2$. Similar to experiment 1, we split $truth\text{-}set_2$ into two parts with 70:30 ratio. $training\text{-}set_2$ is used for hyper-parameter optimization and training, and $testing\text{-}set_2$ is set aside toward final testing[4]. The hyper-parameters, their different configuration options, and best performing selections for experiment 2 are shown in Table VII. Experiments 1 and 2 raise the following research question.

RQ3: How does the IID assumption affect the performance of relation classification model?

VII. Experimental Results

A. Experiment 1 Results

In experiment 1, we compare the labels of $testing\text{-}set_1$ with the predicted relations to answer RQ1 and RQ2. We report the model's performance using precision, recall, and F-1 score. $test\text{-}set_1$ contains 856 direct hypernymy, 87 synonymy, and 19,605 unknown instances. Since our dataset is skewed toward unknown, we opt for F-1 score, which provides a better balance between precision and recall. Table VIII 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- 10 are used to calculate performance measures, which appear in Table IX. 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.

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

TABLE VII: Experiment 2: Parameter Configuration Options

$$Precision_{relation} = \frac{CP_{relation}}{\#Predictions_{relation}} \quad (8)$$

$$Recall_{relation} = \frac{CP_{relation}}{\#Labels_{relation}} \quad (9)$$

$$F\text{-}1_{relation} = 2 \times \frac{Precision_{relation} \times Recall_{relation}}{Precision_{relation} + Recall_{relation}} \quad (10)$$

	Actual Direct Hypernymy	**Actual Synonymy**	**Actual Unknown**	**Total Predictions/-Class**
Predicted Direct Hypernymy	372	35	239	646
Predicted Synonymy	0	0	5	5
Predicted Unknown	84	52	19,361	19,497
Total Labels/-Class	456	87	19,605	

TABLE VIII: Experiment 1: Confusion Matrix

	Hypernymy	**Synonymy**	**Unknown**
Precision	0.575	0.000	0.993
Recall	0.815	0.000	0.987
F-1 Score	0.673	0.000	0.989

TABLE IX: Experiment 1: Performance Measures

The manual ontology construction method requires comparing paired information types by at least two analysts [10], [11]. Therefore, a complete analysis of n information types requires $\frac{n\times(n-1)}{2}$ comparisons. The average time for a pair comparison is reported at 11.72 seconds [13], which makes the task tedious and not feasible for large n. Further, the method is susceptible to human error due to fatigue and gaps in analysts' domain knowledge [13]. In addition, as language use evolves, the ontology would need to be extended over time. Evidence from Bhatia et al. shows that between 23-71% of information types in any new privacy policy will be previously unseen [79]. To address RQ1, we evaluate the extent that the relation classification model can reduce ontology construction effort. The ground-truth ontology contains 367 information types. Using the manual method, an analyst must compare $\frac{367\times(367-1)}{2} = 67,161$ information type pairs during construction (~218 hours). A key challenge is reducing the number of comparisons, particularly of information type pairs that are unknown and that dominate the space of comparisons, e.g., the unknown dominates %97 of the $truth\text{-}set_1$ (see Table II). The 0.987 recall for predicting unknown pairs enables analysts to identify unknown pairs with high confidence, significantly reducing comparison time by 97%.

In experiment 1, $truth\text{-}set_1$ includes direct hypernymy relations and thus label indirect, or ancestor hypernymy relations as unknown. 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. RQ2 investigates the effect of missing transitive hypernymy on relation classification performance. To answer RQ2, we analyze the logical entailment of 1,207 falsely predicted hypernymy relations that were labeled unknown. We utilize the OWL API HermiT reasoner[5] for this analysis. The results show 44% of falsely predicted hypernymy relations with unknown labels are logically entailed through indirect hypernymy. Table X shows

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

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
Player interactions	Interactions	H	U	Yes	2
Mobile devices 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	-

TABLE X: Examples of False Hypernymy Predictions

a sample analysis result, where the model prediction (Pre) is hypernymy (H) and the actual label (Act) is unknown (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.

B. Experiment 2 Results

In experiment 2, we include direct and indirect hypernymy in the actual labeled relations. The model is trained on $testing\text{-}set_2$ containing 2,116 information type pairs labeled as hypernymy (direct and transitive), 106 information type pairs labeled as synonymy, and 17,926 pairs as unknown. We evaluate the performance of the relation classification model by comparing the actual labels of $testing\text{-}set_2$ with the predicted relations.

RQ3 seeks to investigate the effect of transitional hypernymy in classification performance. We use precision, recall, and F-1 score as measures to evaluate the performance on $testing\text{-}set_2$. The confusion matrix for this evaluation is presented in Table XI. In contrast to Table VIII, some portion of unknown labeled instances are shifted toward hypernymy. Hence, the model's performance on hypernymy shows a significant improvement as shown in Table XII.

VIII. Threats to Validity

Construct Validity: Construct validity concerns whether we are measuring what we believe we are measuring. Because we utilize a previously published ontology (See §VI-A) as the ground truth [10], [11], [18], there is a risk that the expected relations are incorrect. The ontology authors describe two

	Actual Hypernymy	**Actual Synonymy**	**Actual Unknown**	**Total Predictions /Class**
Predicted Hypernymy	2,013	58	257	2,328
Predicted Synonymy	0	0	0	0
Predicted Unknown	103	48	17,669	17,820
Total Labels/Class	2,116	106	17,926	

TABLE XI: Experiment 2: Confusion Matrix

	Hypernymy	**Synonymy**	**Unknown**
Precision	0.864	0.000	0.991
Recall	0.951	0.000	0.985
F-1 Score	0.904	0.000	0.985

TABLE XII: Experiment 2: Performance Measures

rounds of consensus-building while developing the ontology, including an inter-rater reliability Kappa statistic of 0.977 and 0.979, which is extremely high above-chance agreement [10].

Internal Validity: We mitigate threats to internal validity by applying established training procedures: the ground-truth in both experiments are partitioned into training and testing sets with 70:30 ratio, and we employ a 10-fold cross validation on the training sets for each experiment. Due to data imbalance, however, the model fails to accurately predict synonymy results. According to Tables II and V, synonymy relations constitute only 0.46% of instances in both $truth\text{-}set_1$ and $truth\text{-}set_2$. To address this threat, we introduce a weighted cross-entropy loss function (see §V-D) in our model. Despite biasing weights for prediction, which enhances hypernymy predictions, synonymy predictions remain unaffected. We recognize that this approach fails to fully mitigate the risk to internal validity. Additional effort is required to reduce the effect of the minority class (i.e., synonymy) by applying over-sampling methods to synonymy instances [80].

External Validity: We acknowledge that the model is only trained on platform information types (see §VI-A), which does not include domain-specific information types, such as health-, finance-, dating-, and shopping-related app data, to name a few. Additional experiments are required to investigate the generalizability of the model.

IX. Discussion

The use of ontologies are directly impactful to novel techniques in privacy-sensitive static and dynamic information flow tracing in software engineering [81], [54], [11]. However, due to technical challenges in information flow tracing, such approaches may not yet be broadly used in industry. That said, continued pressure by regulators and corporate compliance under privacy laws, including the E.U. General Data Protection Regulation, is driving the need to understand where companies collect and use sensitive data. A preliminary step in rationalizing corporate data flows, is classifying stored data by a broad category, which evidence suggests can easily reach into the thousands of distinct types [79]. Without explicit, scalable procedures to construct these ontologies and detect relationships between information type names, companies will produce compliance gaps, where rules for protecting data are applied unevenly due to misalignments arising from ambiguous and vague information type names. Our model for inferring semantic relations between information types can bolster these efforts and increase their viability by providing a general approach for ontology construction, thus yielding a strong indirect benefit to engineers and regulators.

We now examine the characteristics of the ground-truth ontology. This ontology contains 367 information types, resulting in $\frac{367\times(367-1)}{2} = 67,161$ information type pairs that an analyst must compare during construction. According to Table V in experiment 2, $truth\text{-}set_2$ contains 59,781 unknown pairs. Consequently, noting 67,161 possible pairs, 89% of these pairs are identified as unknown in the ground-truth ontology. This percentage reflects that: (1) given all possible information type pair comparisons, there are few relationships (high sparsity); and (2) the corresponding imbalance between related (e.g., hypernymy and synonymy) and unknown pairs is intrinsic to this problem's nature. Given this observation, the model identifies unknown information type pairs with 0.985 F-1 score, greatly reducing the burden of manually checking the unknown pairs in the search space of possible relations for an analyst. In addition, $test\text{-}set_2$ contains 2,222 positive relationships (i.e., direct and transitive hypernymy and synonymy). According to Table XI for experiment 2, the model identifies 2,013 hypernymy from this positive space. The high F-1 score of 0.904 suggests that a requirements analyst can trust the hypernymy relationships inferred by the model.

X. Conclusion

Privacy policies are well-established sources of requirements in software engineering. However, such documents are subject to abstraction and ambiguity, making requirements extraction a challenging task. We focus on the role of hypernyms and their formal relationships among terminology in privacy policies to propose a model for constructing partial ontologies. Such ontologies can be used as knowledge bases by requirements analysts for resolving conflicting interpretations of ambiguous terminology. As improvements to our model, we plan to apply over-sampling methods to synonymy relations. Further, we plan to evaluate ontologies created through our model in privacy detection misalignment tools and privacy question-answering systems. We also plan to automatically extract information types from privacy policies and apply our current model to infer semantic relations. Such an improvement will enhance automatic elicitation of privacy requirements.

Acknowledgment

This research was supported by NSF awards #1736209, #1748109, #1453139, #2007298, #2007718, and #1948244.

References

[1] K. Harris, *Privacy on the Go: Recommendations for the Mobile Ecosystem. California Department of Justice*, 2013.

[2] A. I. Anton and J. B. Earp, "A requirements taxonomy for reducing web site privacy vulnerabilities," *Requirements Engineering*, vol. 9, no. 3, pp. 169–185, 2004.

[3] T. D. Breaux, H. Hibshi, and A. Rao, "Eddy, a formal language for specifying and analyzing data flow specifications for conflicting privacy requirements," *Requirements Engineering*, vol. 19, no. 3, pp. 281–307, 2014.

[4] A. K. Massey, J. Eisenstein, A. I. Antón, and P. P. Swire, "Automated text mining for requirements analysis of policy documents," in *2013 21st IEEE International Requirements Engineering Conference (RE)*. IEEE, 2013, pp. 4–13.

[5] A. K. Massey, E. Holtgrefe, and S. Ghanavati, "Modeling regulatory ambiguities for requirements analysis," in *International Conference on Conceptual Modeling*. Springer, 2017, pp. 231–238.

[6] E. Kamsties and B. Peach, "Taming ambiguity in natural language requirements," in *Proceedings of the Thirteenth International Conference on Software and Systems Engineering and Applications*, 2000.

[7] J. R. Reidenberg, J. Bhatia, T. D. Breaux, and T. B. Norton, "Ambiguity in privacy policies and the impact of regulation," *The Journal of Legal Studies*, vol. 45, no. S2, pp. S163–S190, 2016.

[8] D. M. Berry and E. Kamsties, "Ambiguity in requirements specification," in *Perspectives on software requirements*. Springer, 2004, pp. 7–44.

[9] A. K. Massey, R. L. Rutledge, A. I. Antón, and P. P. Swire, "Identifying and classifying ambiguity for regulatory requirements," in *2014 IEEE 22nd International Requirements Engineering Conference (RE)*. IEEE, 2014, pp. 83–92.

[10] M. B. Hosseini, S. Wadkar, T. D. Breaux, and J. Niu, "Lexical similarity of information type hypernyms, meronyms and synonyms in privacy policies," in *AAAI Fall Symposium*, 2016.

[11] R. Slavin, X. Wang, M. B. Hosseini, J. Hester, R. Krishnan, J. Bhatia, T. D. Breaux, and J. Niu, "Toward a framework for detecting privacy policy violations in Android application code," in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 25–36.

[12] J. Bhatia, T. D. Breaux, J. R. Reidenberg, and T. B. Norton, "A theory of vagueness and privacy risk perception," in *2016 IEEE 24th International Requirements Engineering Conference (RE)*. IEEE, 2016, pp. 26–35.

[13] M. B. Hosseini, T. D. Breaux, and J. Niu, "Inferring ontology fragments from semantic role typing of lexical variants," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2018, pp. 39–56.

[14] M. C. Evans, J. Bhatia, S. Wadkar, and T. D. Breaux, "An evaluation of constituency-based hyponymy extraction from privacy policies," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 2017, pp. 312–321.

[15] M. B. Hosseini, R. Slavin, T. Breaux, X. Wang, and J. Niu, "Disambiguating requirements through syntax-driven semantic analysis of information types," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2020, pp. 97–115.

[16] T. D. Breaux and D. L. Baumer, "Legally "reasonable" security requirements: A 10-year FTC retrospective," *Computers & Security*, vol. 30, no. 4, pp. 178–193, 2011.

[17] FTC. (2019) FTC's $5 billion facebook settlement: Record-breaking and history-making.

[18] R. Slavin, X. Wang, M. B. Hosseini, J. Hester, R. Krishnan, J. Bhatia, T. D. Breaux, and J. Niu, "Pvdetector: a detector of privacy-policy violations for Android apps," in *2016 IEEE/ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft)*. IEEE, 2016, pp. 299–300.

[19] "IEEE Recommended Practice for Software Requirements Specifications," in *IEEE Computer Society. Software Engineering Standards Committee and IEEE-SA Standards Board*, vol. 830, no. 1998. IEEE, 1998.

[20] R. Harwell, E. Aslaksen, R. Mengot, I. Hooks, and K. Ptack, "What is a requirement?" in *INCOSE International Symposium*, vol. 3, no. 1. Wiley Online Library, 1993, pp. 17–24.

[21] D. Jurafsky and J. H. Martin, *Speech and language processing*. Pearson London, 2014, vol. 3.

[22] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[23] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[24] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[25] Y. LeCun *et al.*, "Generalization and network design strategies," *Connectionism in perspective*, pp. 143–155, 1989.

[26] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[27] G. Lami, S. Gnesi, F. Fabbrini, M. Fusani, and G. Trentanni, "An automatic tool for the analysis of natural language requirements," *Informe técnico, CNR Information Science and Technology Institute, Pisa, Italia, Setiembre*, 2004.

[28] A. Nigam, N. Arya, B. Nigam, and D. Jain, "Tool for automatic discovery of ambiguity in requirements," *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 5, p. 350, 2012.

[29] N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry, "Requirements for tools for ambiguity identification and measurement in natural language requirements specifications," *Requirements Engineering*, vol. 13, no. 3, pp. 207–239, 2008.

[30] A. Ferrari and S. Gnesi, "Using collective intelligence to detect pragmatic ambiguities," in *2012 20th IEEE International Requirements Engineering Conference (RE)*. IEEE, 2012, pp. 191–200.

[31] A. Ferrari, P. Spoletini, and S. Gnesi, "Ambiguity and tacit knowledge in requirements elicitation interviews," *Requirements Engineering*, vol. 21, no. 3, pp. 333–355, 2016.

[32] S. Boyd, D. Zowghi, and V. Gervasi, "Optimal-constraint lexicons for requirements specifications," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2007, pp. 203–217.

[33] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[34] D. Fensel, D. McGuiness, E. Schulten, W. K. Ng, G. P. Lim, and G. Yan, "Ontologies and electronic commerce," *IEEE Intelligent Systems*, vol. 16, no. 1, pp. 8–14, 2001.

[35] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E. Chioasca, and R. T. Batista-Navarro, "Natural language processing (NLP) for requirements engineering: A systematic mapping study," *arXiv preprint arXiv:2004.01099*, 2020.

[36] D. Dermeval, J. Vilela, I. I. Bittencourt, J. Castro, S. Isotani, P. Brito, and A. Silva, "Applications of ontologies in requirements engineering: a systematic review of the literature," *Requirements Engineering*, vol. 21, no. 4, pp. 405–437, 2016.

[37] D. G. Gordon and T. D. Breaux, "Reconciling multi-jurisdictional legal requirements: A case study in requirements water marking," in *2012 20th IEEE International Requirements Engineering Conference (RE)*. IEEE, 2012, pp. 91–100.

[38] K. K. Breitman and J. C. S. do Prado Leite, "Ontology as a requirements engineering product," in *Proceedings. 11th IEEE International Requirements Engineering Conference (RE), 2003*. IEEE, 2003, pp. 309–319.

[39] A. Oltramari, D. Piraviperumal, F. Schaub, S. Wilson, S. Cherivirala, T. B. Norton, N. C. Russell, P. Story, J. Reidenberg, and N. Sadeh, "Privonto: A semantic framework for the analysis of privacy policies," *Semantic Web*, vol. 9, no. 2, pp. 185–203, 2018.

[40] L. Humphreys, G. Boella, L. van der Torre, L. Robaldo, L. Di Caro, S. Ghanavati, and R. Muthuri, "Populating legal ontologies using semantic role labeling," *Artificial Intelligence and Law*, pp. 1–41, 2020.

[41] X. Wang, X. Qin, M. B. Hosseini, R. Slavin, T. D. Breaux, and J. Niu, "Guileak: Tracing privacy policy claims on user input data for Android applications," in *Proceedings of the 40th International Conference on Software Engineering*, 2018, pp. 37–47.

[42] R. Snow, D. Jurafsky, and A. Y. Ng, "Learning syntactic patterns for automatic hypernym discovery," in *Advances in neural information processing systems*, 2005, pp. 1297–1304.

[43] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *Proceedings of the 14th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, 1992, pp. 539–545.

[44] R. C. Bunescu and R. 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*. Association for Computational Linguistics, 2005, pp. 724–731.

[45] M. Mintz, S. Bills, R. Snow, and D. 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*. Association for Computational Linguistics, 2009, pp. 1003–1011.

[46] D. Zelenko, C. Aone, and A. Richardella, "Kernel methods for relation extraction," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1083–1106, 2003.

[47] Z. GuoDong, S. Jian, Z. Jie, and Z. Min, "Exploring various knowledge in relation extraction," in *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2005, pp. 427–434.

[48] D. Zeng, K. Liu, S. Lai, G. Zhou, J. Zhao *et al.*, "Relation classification via convolutional deep neural network." in *COLING*, 2014, pp. 2335–2344.

[49] P. Zhou, W. Shi, J. Tian, Z. Qi, B. Li, H. Hao, and B. 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)*, vol. 2, 2016, pp. 207–212.

[50] P. D. Turney and S. M. Mohammad, "Experiments with three approaches to recognizing lexical entailment," *Natural Language Engineering*, vol. 21, no. 3, pp. 437–476, 2015.

[51] M. Rei, D. Gerz, and I. Vulić, "Scoring lexical entailment with a supervised directional similarity network," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 638–643. [Online]. Available: https://aclanthology.org/P18-2101

[52] M. Nickel and D. Kiela, "Learning continuous hierarchies in the lorentz model of hyperbolic geometry," in *International Conference on Machine Learning*. PMLR, 2018, pp. 3779–3788.

[53] C. Wang and X. He, "Birre: learning bidirectional residual relation embeddings for supervised hypernymy detection," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 3630–3640.

[54] H. Harkous, K. Fawaz, R. Lebret, F. Schaub, K. G. Shin, and K. Aberer, "Polisis: Automated analysis and presentation of privacy policies using deep learning," in *27th USENIX Security Symposium*, 2018, pp. 531–548.

[55] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[56] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[57] W. Ling, C. Dyer, A. W. Black, and I. 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*, 2015, pp. 1299–1304.

[58] D. Tang, B. Qin, and T. 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)*, 2015, pp. 1014–1023.

[59] N. Viennot, E. Garcia, and J. Nieh, "A measurement study of google play," in *The 2014 ACM international conference on Measurement and modeling of computer systems*, 2014, pp. 221–233.

[60] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proceedings of the 2015 conference on empirical methods in natural language processing*, 2015, pp. 1422–1432.

[61] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint arXiv:1503.00075*, 2015.

[62] J. Guo, J. Cheng, and J. Cleland-Huang, "Semantically enhanced software traceability using deep learning techniques," in *Proceedings of the 39th International Conference on Software Engineering*. IEEE Press, 2017, pp. 3–14.

[63] G. Namata, P. Sen, M. Bilgic, L. Getoor, M. Sahami, and A. Srivastava, "Collective classification for text classification," *Text Mining*, pp. 51–69, 2009.

[64] G. M. Namata, S. Kok, and L. Getoor, "Collective graph identification," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 87–95.

[65] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," *Requirements Engineering*, vol. 21, no. 3, pp. 311–331, 2016.

[66] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? on automatically classifying app reviews," in *2015 IEEE 23rd international requirements engineering conference (RE)*. IEEE, 2015, pp. 116–125.

[67] G. Williams and A. Mahmoud, "Mining twitter feeds for software user requirements," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 2017, pp. 1–10.

[68] E. Guzman, R. Alkadhi, and N. Seyff, "A needle in a haystack: What do twitter users say about software?" in *2016 IEEE 24th International Requirements Engineering Conference (RE)*. IEEE, 2016, pp. 96–105.

[69] ——, "An exploratory study of twitter messages about software applications," *Requirements Engineering*, vol. 22, no. 3, pp. 387–412, 2017.

[70] E. Guzman, M. Ibrahim, and M. Glinz, "Prioritizing user feedback from twitter: A survey report," in *2017 IEEE/ACM 4th International Workshop on CrowdSourcing in Software Engineering (CSI-SE)*. IEEE, 2017, pp. 21–24.

[71] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, "Automated classification of non-functional requirements," *Requirements engineering*, vol. 12, no. 2, pp. 103–120, 2007.

[72] ——, "The detection and classification of non-functional requirements with application to early aspects," in *14th IEEE International Requirements Engineering Conference (RE'06)*. IEEE, 2006, pp. 39–48.

[73] I. Hussain, L. Kosseim, and O. Ormandjieva, "Using linguistic knowledge to classify non-functional requirements in srs documents," in *International Conference on Application of Natural Language to Information Systems*. Springer, 2008, pp. 287–298.

[74] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[75] S. Chakrabarti, B. Dom, and P. Indyk, "Enhanced hypertext categorization using hyperlinks," *Acm Sigmod Record*, vol. 27, no. 2, pp. 307–318, 1998.

[76] J. Neville and D. Jensen, "Iterative classification in relational data," in *Proc. AAAI-2000 workshop on learning statistical models from relational data*, 2000, pp. 13–20.

[77] D. Jensen, J. Neville, and B. Gallagher, "Why collective inference improves relational classification," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004, pp. 593–598.

[78] F. Baader, D. Calvanese, D. McGuinness, P. Patel-Schneider, D. Nardi *et al.*, *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.

[79] J. Bhatia, T. D. Breaux, and F. Schaub, "Mining privacy goals from privacy policies using hybridized task recomposition," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 25, no. 3, pp. 1–24, 2016.

[80] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[81] S. Zimmeck and S. M. Bellovin, "Privee: An architecture for automatically analyzing web privacy policies," in *23rd USENIX Security Symposium*, 2014, pp. 1–16.