POLIGRAPH: Automated Privacy Policy Analysis using Knowledge Graphs

Hao Cui, Rahmadi Trimananda, Athina Markopoulou, Scott Jordan

University of California, Irvine

Abstract

Privacy policies disclose how an organization collects and handles personal information. Recent work has made progress in leveraging natural language processing (NLP) to automate privacy policy analysis and extract data collection statements from different sentences, considered in isolation from each other. In this paper, we view and analyze, for the first time, the entire text of a privacy policy in an integrated way. In terms of methodology: (1) we define POLIGRAPH, a type of knowledge graph that captures statements in a privacy policy as relations between different parts of the text; and (2) we develop an NLP-based tool, POLIGRAPH-ER, to automatically extract POLIGRAPH from the text. In addition, (3) we revisit the notion of ontologies, previously defined in heuristic ways, to capture subsumption relations between terms. We make a clear distinction between local and global ontologies to capture the context of individual privacy policies, application domains, and privacy laws. Using a public dataset for evaluation, we show that POLIGRAPH-ER identifies 40% more collection statements than prior state-of-the-art, with 97% precision. In terms of applications, POLIGRAPH enables automated analysis of a corpus of privacy policies and allows us to: (1) reveal common patterns in the texts across different privacy policies, and (2) assess the correctness of the terms as defined within a privacy policy. We also apply POLIGRAPH to: (3) detect contradictions in a privacy policy, where we show false alarms by prior work, and (4) analyze the consistency of privacy policies and network traffic, where we identify significantly more clear disclosures than prior work.

1 Introduction

Privacy Policies. Privacy laws, such as the General Data Protection Regulation (GDPR) [1], the California Consumer Privacy Act (CCPA) [2], and other data protection laws, require organizations to disclose the personal information they collect, as well as how and why they use and share it. Privacy policies are the primary legally-binding way for organizations to disclose their data collection practices to the users of their products. They receive much attention from many stakeholders, such as users who want to exercise their rights, developers who want their systems to be compliant with privacy laws, and law enforcement agencies who want to audit organizations' data collection practices and hold them accountable. Unfortunately, privacy policies are typically lengthy and complicated, making it hard not only for the average user to understand, but also for experts to analyze in depth and at scale [3].

NLP Analysis and Limitations. To address this challenge, as well as to facilitate expert analysis [7] and crowdsourced annotation [8], the research community has recently applied natural language processing (NLP) to automate the analysis of privacy policies. State-of-the-art examples include the following: PolicyLint [9] extracts data types and entities that collect them, and analyzes potential contradictions within a privacy policy; PoliCheck [10] builds on PolicyLint and further compares the privacy policy statements with the data collection practices observed in the network traffic; Polisis [11] and PurPliance [5] extract data collection purposes; and OVRseen [6] leverages PoliCheck and Polisis to associate data types, entities, and purposes. Despite promising results, this body of work also has certain limitations.

First, existing privacy policy analyzers extract statements (about what is collected, *i.e.,* data type; who collects it, *i.e.,* entity; and for what purpose) as disconnected labels [11] or tuples [5,9], ignoring the links between information disclosed across sentences, paragraphs or sections. However, today's privacy policies typically have a structure that discloses data types being collected, third-party sharing and usage purposes in separate sections[1], as shown in the example in Figure 1(a). Polisis [11] uses separate text classifiers to label data types, third-party entities and purposes disclosed in each paragraph. Without connecting these labels, it is unclear which data type is collected by which entity, and what purpose applies. PolicyLint [9] and PurPliance [5] adopt tuple representations

[1]We read through 200 privacy policies in our test set (see Section 4). Among them, 135 discuss definitions and practices concerning the same data types in different sections, requiring to put the information together to get the full context about collection, use and sharing of these data types. In particular, 104 divide content into sections addressing collection, use, and sharing of "personal information", resembling the structure shown in Figure 1(a).

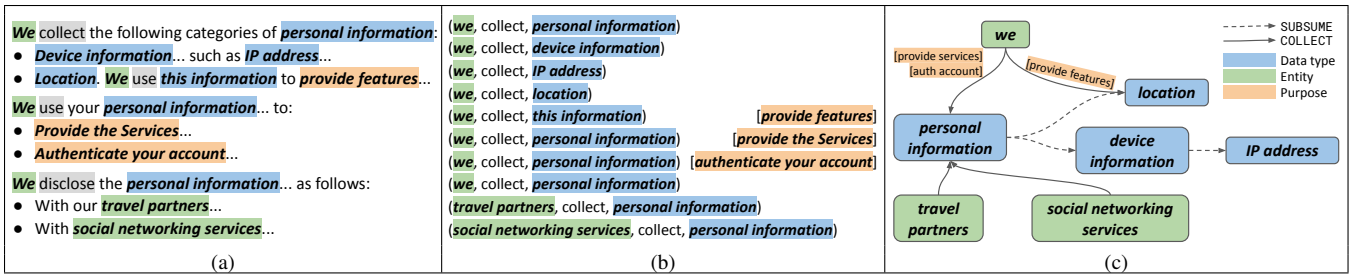


Figure 1: **Example of a privacy policy and analysis approaches. (a)** The excerpt is from the policy of KAYAK [4]. It contains sections and lists, regarding: what is collected (data type), how it is used (purpose), who receives the information (entity), and references across sentences (*e.g.,* "personal information" relates to other data types; "this information" refers to "location"). **(b)** Prior work extracts elements found in each sentence, mainly data types and entities, as disconnected tuples. Purposes can also be extracted to extend the tuple [5, 6]. **(c)** POLIGRAPH is a knowledge graph that encodes data types, entities, and purposes; and two types of relations between them (collection and subsumption), possibly specified across different sentences. A *COLLECT* edge represents that a data type is collected by an entity, while edge attributes represent the purposes of that collection. *SUBSUME* edges represent the subsumption relations between generic and specific terms.

that put together entities, data types and purposes disclosed in each sentence, as shown in Figure 1(b). However, the tuples still miss context from other sentences. For example, it cannot be inferred from the tuples that the purpose "provide features" applies to the collection of "location"; or that the usage purposes and third-party entities in later sections are related to the specific types of "personal information" (*e.g.,* "device information") listed in the first section.

Second, because of this incomplete context, prior work needs to map and relate the semantics of the terms across different sentences by introducing *ontologies* that encode subsumption relations between data types or entities. So far, these ontologies have been built in a manual or semi-automated fashion by domain experts, who define lists of terms commonly found in privacy policy text and other sources (*e.g.,* network traffic), and subsumption relations between them (*e.g.,* the term "device information" subsumes "IP address"). The resulting ontologies are not universal: they do not necessarily agree with all privacy policies and need to be adapted to different application domains, *e.g.,* mobile [5, 9, 10], smart speakers [12, 13], and VR [6]. As a result, they often generate ambiguous or wrong results that require further validation by experts. Manandhar *et al.* [14] recently reported that state-of-the-art analyzers [9–11] incorrectly reason about more than half of the privacy policies they analyzed.

The POLIGRAPH Framework. Our key observation is that a policy[2] should be treated in its entirety, leveraging terms in different sentences that are related. To that end, we make the following methodological contributions.

First, we propose to extract and encode statements in a policy (*i.e.,* what *data types* are collected, with what *entities* they are shared, and for what *purposes*) into a knowledge graph [15, 16], which we refer to as POLIGRAPH; Figure 1(c) shows an example [3]. Nodes represent data types or entities. Edges represent relations between nodes, *e.g.,* an entity may *collect* a particular data type, and a more generic data type may *subsume* a more specific data type. An edge representing data collection may have an attribute indicating the purposes. The graph in Figure 1(c) naturally links the extracted information by merging the same data types and entities and establishing edges between them. It allows inferences such as "IP address" being collected for the purpose "provide services", and "location" being collected by "travel partners".

Second, for policies that are not well written, the extracted POLIGRAPH may be missing subsumption relations between terms that are not fully defined in the policies. To supplement the missing relations, we use ontologies, as in prior work [5, 9, 10]; however, we redefine and use them as follows. First, we consider the subsumption relations extracted from each individual policy as the *local ontology* definied by it. Next, we also define additional subsumption relations that encode external knowledge, beyond what is stated in the text of an individual policy; we refer to these as *global ontologies*. They can be defined by domain experts, using information from multiple policies, or from privacy laws; for example, in Section 2.2, we define a data ontology based on the CCPA [2].

Third, we present POLIGRAPH-ER, a methodology and implementation that applies NLP linguistic analysis to automatically extract and build a POLIGRAPH from the privacy policy text. To that end, we address several challenges, including coreference resolution, list parsing, phrase normalization, and purpose phrase classification, to extract and link more information than previously possible.

Evaluation. We evaluate POLIGRAPH-ER on a public dataset from PoliCheck [10], consisting of over 6K policies from over 13K mobile apps. Our manual validation shows that POLIGRAPH improves the recall of collection statements from 30% to 70%, compared to prior work [9], with over 90% precision.

[2] In the rest of the paper, we refer to a privacy policy simply as "policy".

[3] Examples of *COLLECT* edges, representing collection of a data type by an entity: (1) "we" (first party, *i.e.,* KAYAK) collect "personal information", with the purposes "provide services" and "authenticate your account"; (2) "travel partners" collect (or precisely, are disclosed with) "personal information". Example of *SUBSUME* edges, representing subsumption relations: "personal information" subsumes "location" and "device information", which in turn subsumes "IP address".

The improvement is enabled by both the improved NLP techniques and the knowledge graph representation, which can analyze statements spanning multiple sentences and sections.

Applications. PoliGraph enables two new types of automated analyses, which were not previously possible. First, PoliGraph is used to *summarize policies* in our dataset and reveal common patterns across them. This is possible because PoliGraph, by representing each policy as a whole, allows inferences about more collection statements. We find that 64% of policies disclose the collection of software identifiers and, in particular, cookies. Advertisers and analytics providers are major entities that collect such data. This is further reinforced by the finding that more than half of the policies disclose data usage for non-core purposes, namely for advertising and analytics. We also find that the use of generic terms for data types (*e.g.,* "personal information"), often without more precise definitions, reduces the transparency and leaves the specific data types being collected unknown. Second, different policies may have different definitions of the same terms. By clearly separating local ontologies from global ones, PoliGraph allows us to *assess the correctness* of the term definitions. For example, we find that many policies declare the collected data as "non-personal information", which contradicts common knowledge and our CCPA-based global data ontology (see Sections 2.2 and 5.2). We also find that non-standard terms are widely used, with varied definitions across policies.

We also apply PoliGraph to revisit two known applications of policy analysis. First, to identify *contradictions* within a policy, we extend PoliGraph to analyze negative statements and take into account additional contexts that are crucial for interpreting contradictions, such as (1) fine-grained actions (*e.g.,* "sell" for profit *vs.* "sharing"), and (2) data subjects (*e.g.,* children *vs.* general users). We show that the majority of contradictions found by prior work are false alarms due to language nuances and missing contexts (*e.g.,* data subjects). Second, we apply PoliGraph to analyze *data flow-to-policy consistency*. As a result of the improved recall of our approach, we show that prior work [10] has underestimated the number of policies that clearly disclose some sensitive data flows.

Overview. The rest of the paper is structured as follows. Section 2 defines the proposed PoliGraph framework and the ontologies used with it. Section 3 describes the implementation of PoliGraph-ER that uses NLP to build PoliGraph from the text of a policy. Section 4 presents the evaluation of our framework. Section 5 presents applications of PoliGraph to policy analysis. Section 6 discusses related work. Finally, Section 7 concludes the paper.

2 The PoliGraph Framework

In this section, we introduce PoliGraph, our proposed representation of the entire text of a policy as a knowledge graph. We also revisit the related notion of ontologies, and we propose a new definition and use it with PoliGraph.

2.1 Defining PoliGraph

We define PoliGraph as a knowledge graph that captures statements in a policy considered as a whole. Throughout this section, we will use Figure 1 as our running example to illustrate the terminology and definitions.

Privacy laws, such as the GDPR [1] and the CCPA [2], require that organizations disclose their practices regarding data collection, sharing and use in their policies. To capture these three aspects of disclosures in the policy, we represent the corresponding three kinds of terms in PoliGraph: what *data types* are collected, with what *entities* they are shared, and for what *purposes* they are used.

- *Data type:* This kind of terms refers to the type of data being collected. In Figure 1(a), "location" is a specific collected data type. Generic terms can be used as well, *e.g.,* "personal information" and "device information".
- *Entity:* This kind of terms refers to the organization that receives the collected data. It can be the first party if it is the developer of the product (*e.g.,* website, mobile app, *etc.*) that writes the policy, namely "we" in Figure 1(a); or, otherwise, a third party such as "travel partners" in Figure 1(a).
- *Purpose:* Policies may also specify purposes.[4] In Figure 1(a), purposes include "provide services", "authenticate your account", and "provide features".

In PoliGraph, we represent data types and entities as two different types of nodes. Furthermore, we encode the following relations between them as edges.

- *COLLECT edge*: An entity n may collect a data type d. In Figure 1(a), "personal information" is collected by the first-party entity "we", but it is also shared with the entity "travel partners" (a third party). More formally, a *COLLECT* edge $e_c = n \xrightarrow{\text{COLLECT}} d$ between an entity n and a data type d represents that d is collected by n, namely $collect(n,d)$.
- *SUBSUME edge*: A generic term (*hypernym*) may subsume a more specific term (*hyponym*). For example, "personal information" subsumes "device information" and "location", and "device information" in turn subsumes "IP address". More formally, a *SUBSUME* edge $e_s = hyper \xrightarrow{\text{SUBSUME}} hypo$ connects nodes *hyper* and *hypo*, where both nodes *hyper*, *hypo* are data types or both are entities, and it represents that the more generic term *hyper* subsumes the more specific term *hypo*, namely $subsume(hyper, hypo)$.
- *Purposes as edge attributes*: We represent purposes by assigning them as a list of attributes $Purposes(e_c) = \{p_1, p_2, ...\}$ to each *COLLECT* edge e_c. This is a natural choice that fits how policies are written: one or more purposes are typically associated with a data type and an entity. In Figure 1(a), entity "we" (*i.e.,* KAYAK) collects "this information", which refers to "location", for the purpose "to provide features".

[4] In this paper, we refer to *purposes* of processing of personal data as specified in the GDPR, namely the purposes of collection, use, and sharing. US laws often distinguish among the three, *e.g.,* the CCPA appears to require a policy to separately disclose the purposes of collection / use and the purposes of sharing personal information.

The purpose "to provide features" is captured in the list of attributes $Purposes(e_c) = \{provide\ features\}$ assigned to the *COLLECT* edge $e_c = we \xrightarrow{\text{COLLECT}} location$.

In summary, we define POLIGRAPH, representing knowledge about data collection, sharing and use disclosed within a particular policy, as follows.

Definition 2.1. POLIGRAPH. A POLIGRAPH $G = \langle D, N; E_S, E_C; P \rangle$ is a directed acyclic graph. Each node represents a term that is either a data type $d \in D$ or an entity $n \in N$. Each edge can be either a *SUBSUME* edge $e_s \in E_S$, or a *COLLECT* edge $e_c \in E_C$ as defined above. A *COLLECT* edge e_c has a list of attributes $Purposes(e_c) = \{p_1, p_2, ...\}$, where $p_i \in P$.

Figure 1(c) shows the POLIGRAPH representation of the policy text in Figure 1(a). The technical details about building the graph from verbatim text, such as how to map the co-reference term "this information" to "location", are provided in Section 3. Next, we define relations that can be inferred from POLIGRAPH about policy text.

Definition 2.2. ***Subsumption Relation.*** In a POLIGRAPH G, we say that a term t_1 (hypernym) *subsumes* another term t_2 (hyponym), denoted as $subsume(t_1, t_2)$, *iff* there exists a path from t_1 to t_2 in G where every edge is a *SUBSUME* edge.[5]

Definition 2.3. ***Collection Relation.*** In a POLIGRAPH G, we say an entity $n \in N$ *collects* a data type $d \in D$, denoted as $collect(n, d)$, *iff* there exists an entity $n' \in N$ and a data type $d' \in D$ where $subsume(n', n) \wedge subsume(d', d)$[6] and the edge $n' \xrightarrow{\text{COLLECT}} d'$ exists in G.

Definition 2.4. ***Set of Purposes.*** Following Definition 2.3, if a purpose $p \in Purposes(n' \xrightarrow{\text{COLLECT}} d')$, we say n collects d for the purpose p. We denote the set of all instances of such p in G as a set $purposes(n, d)$.

Beyond what is captured by individual nodes, edges, and attributes, the strength of POLIGRAPH is that it allows us to make inferences. In Figure 1(c), there is no direct edge from "travel partners" to "location", but we can still infer that "location" may be shared with "travel partners" and "social network services". Furthermore, we can also infer that *collect*(*we*, *location*) and *purposes*(*we*, *location*) = {*provide features*}. Such data practices that are implied, but not explicitly stated, would be missed by prior work that only processes individual sentences, and possibly by human readers as well.

Prior state-of-the-art work would have extracted a list of tuples, as depicted in the example of Figure 1(b). PolicyLint [9] and follow-up works [10, 12] extract 2-tuples: ⟨*entity, data type*⟩. Purposes can be extracted independently and appended to form a longer 3-tuple ⟨*entity, data type, purpose*⟩ as in OVRseen [6], or put in a nested tuple as in PurPliance [5]. In all cases, those tuples are extracted from individual sentences that are disconnected from each other. As a result, prior work would fail to identify implied statements. In contrast, POLIGRAPH connects terms with the same semantics in different sentences, allowing inferences and improving coverage.

Another major strength of POLIGRAPH is that its modular design makes it easy to extend to capture additional relations. In Section 5.3, we present POLIGRAPH extensions to handle finer-grained semantics, including negative edges and subtypes of *COLLECT* edges to distinguish among data actions (*e.g.,* "sell" for profit *vs.* "sharing"), as well as data subjects.

2.2 Ontologies

Policies refer to data types and entities at different semantic granularities. For example, "device information" in Figure 1(a) is a generic data type that subsumes "IP address" and maybe other more specific data types. Prior work [6, 9, 10] has introduced hierarchies of terms, namely ontologies, to define the subsumption relations between data types or entities. They typically define the data and entity ontologies heuristically and manually, by considering a combination of information found in the network traffic and in the policy text, as well as using domain expertise to organize terms into hierarchies.

We revisit the notion of ontologies under the POLIGRAPH framework. First, POLIGRAPH naturally captures subsumption relations described in an individual policy, which form the *local ontology*. Ideally, if a policy is written in a clear and complete way, it should either use specific terms, or clearly define generic terms that will be captured by the corresponding local ontology. In practice, policies are not perfectly written and parts of the ontology may be missing. For example, in Figure 1(a), the term "social networking services" is not further explained. Furthermore, some policies may provide misleading definitions, *e.g.,* "geolocation" is described as non-personal information, whereas it is widely considered personal by the public and privacy laws (see Section 5.2). Second, we define and design *global ontologies* that encode external knowledge or ground truth, as in prior work. For the first time, the distinction between local and global ontologies provides a principled way to summarize an individual policy, as well as to assess the completeness and correctness of definitions by comparing the local against the global ontologies.

2.2.1 Local Ontologies

In POLIGRAPH, *SUBSUME* edges between data types or entities induce a directed acyclic graph, which we refer to as a local ontology, capturing the relations between more generic and more specific terms, as defined within a particular policy. We define local data and entity ontologies as follows.

Definition 2.5. ***Local Ontology.*** A local ontology is either a data ontology $o_d = \langle D, E_d \rangle$ or an entity ontology $o_n = \langle N, E_n \rangle$, a directed acyclic graph that is a subgraph of POLIGRAPH $G = \langle D, N; E_S, E_C; P \rangle$, in which every node is a data type $d \in D$ or an entity $n \in N$, and every edge $e_d \in E_d, e_n \in E_n$ (where $E_d, E_n \subset E_S$) is a *SUBSUME* edge.

[5] A subsumption relation is naturally transitive. To simplify other definitions, we also make it reflexive, *i.e.,* every term subsumes itself.

[6] That is, a policy may disclose data collection using generic terms. For instance, in Figure 1(c), we have *collect*(*we*, *IP address*) because "IP address" is also "personal information".

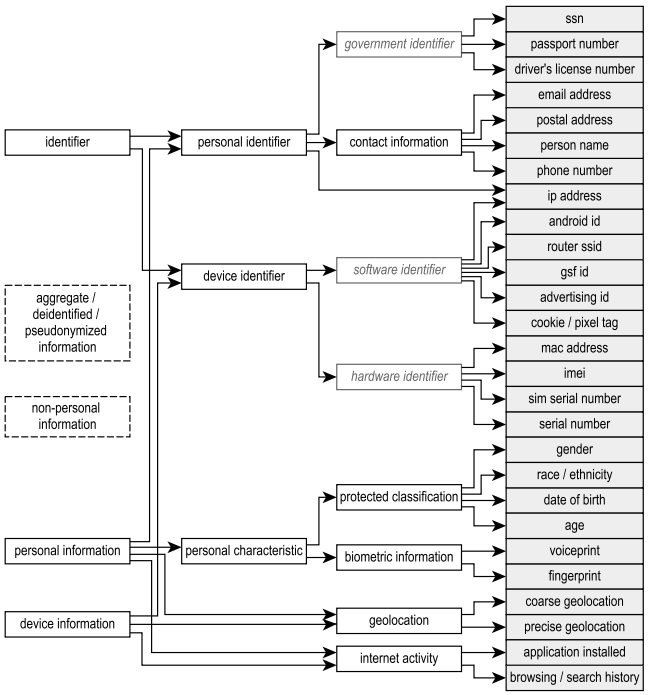


Figure 2: Global Data Ontology based on the CCPA.

In Figure 1(c), the four blue nodes containing data types form the local data ontology: the root node is "personal information" and the leaf nodes are "location" and "IP address". The local entity ontology, which contains the three green nodes, does not have a nontrivial hierarchical structure because the policy does not further explain the terms "travel partners" and "social networking services".

2.2.2 Global Ontologies

We define a global ontology to encode external knowledge, *i.e.,* outside a particular policy, which we consider as ground truth in that context. It provides a reference against which we can compare and evaluate individual policies, as well as a complement to missing definitions in policies.

Definition 2.6. ***Global Ontology.*** A global ontology is either a data ontology $O_d = \langle D_d, E_d \rangle$ or an entity ontology $O_n = \langle N_n, E_n \rangle$ that is a directed acyclic graph, where every node is a data type $d \in D_d$ or an entity $n \in N_n$, and every edge $e_d \in E_d$ or $e \in E_n$ is a *SUBSUME* edge.

Prior work [6,9,10] has implicitly and heuristically defined such global ontologies, by taking into account and combining the union of all subsumption relations extracted from policies in their corpus, and the data types and entities observed in the actual system's output (*e.g.,* network traffic). However, such global ontologies have not been universal: they may include subjective judgment, and they typically do not apply across application domains. For example, PoliCheck's data ontology does not assume "personal information" to include "device information": this contradicts the content of the policy depicted in Figure 1(a). Although we recognize that there is no single way to define perfect global ontologies, we propose

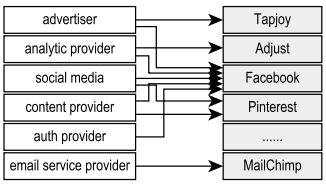


Figure 3: Global Entity Ontology based on [17, 18].

that we rely on authoritative sources, such as privacy laws, to define them. An example is described next, but other designs can be used with POLIGRAPH as well.

Global Data Ontology Based on the CCPA. As a concrete, illustrative example, we propose a global data ontology that is based on the CCPA [2]. The CCPA governs the collection, use, and sharing of personal information, as defined therein, by companies that do business in California. To build the CCPA-based global data ontology, we start with the definition of "personal information" in CCPA Section 1798.140(v)(1), which includes, but is not limited to, specific data types, including a person's name, social security number, postal address, email address, and IP address. We place such specific data types into the ontology as leaf nodes. Then, since policies often disclose the collection of *categories* of these specific data types, *e.g.,* "contact information" instead of "email address" and "postal address", we organize these specific data types into categories delineated by subsumption relations. The CCPA's definition of personal information also includes categories for which it does not list specific data types, *e.g.,* "biometric information". In such cases, we include the categories in the global data ontology and augment it with common specific data types, *e.g.,* "biometric information" includes "voiceprint" and "fingerprint". Similarly, the CCPA uses the term "device identifier" but does not define it, while we include it as a category in the global data ontology, and place specific device identifiers in that category. Figure 2 shows the CCPA-based global data ontology. The above is meant as a concrete example of a global ontology based on a privacy law. Different laws (*e.g.,* GDPR) can lead to different global ontologies.

Global Entity Ontology. Privacy laws give examples of the types of entities, but not the exhaustive list of entities, with whom an organization may share personal information. We follow policies that often categorize entities by service types. We obtain a list of entities and their categories from the DuckDuckGo Tracker Radar dataset [17] and a CrunchBase-based dataset [18]. Based on these sources, containing 4,709 entities in total, we propose a simple two-level ontology that classifies entities into six categories as shown in Figure 3.

The global ontologies, serving as the ground truth of subsumption relations, are used to categorize unorganized data types and entities in POLIGRAPHs (see Section 5.1), assess the correctness of term definitions in individual policies (see Section 5.2), as well as complement term definitions in case of missing definitions when we check vague disclosures of data flows (see Section 5.4).

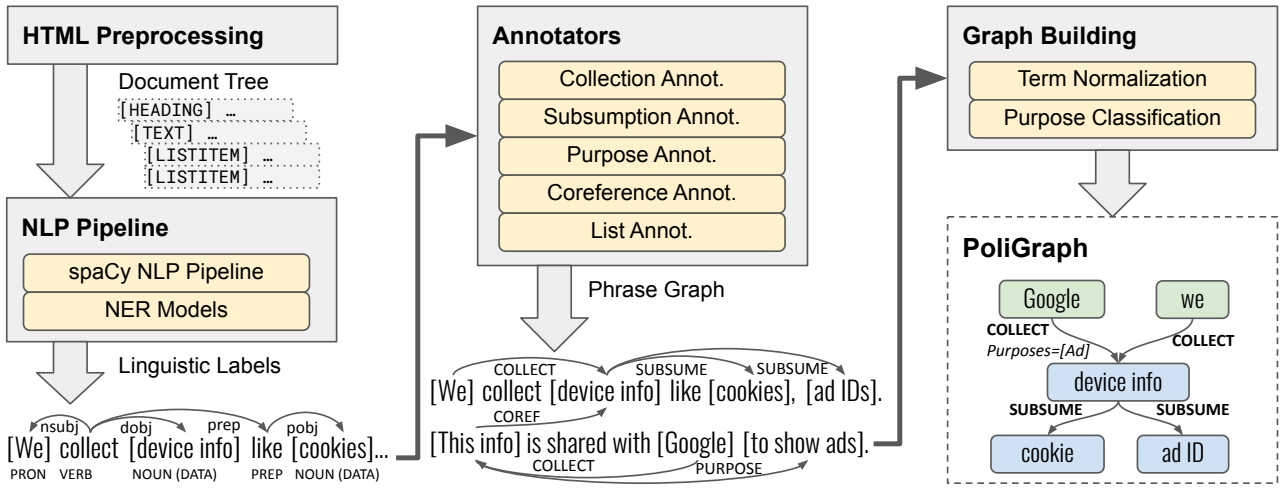


Figure 4: **Overview of PoliGraph-er implementation.** First, PoliGraph-er preprocesses the HTML document to produce a simplified document tree structure. Second, the NLP pipeline takes the document tree and labels sentences with linguistic labels. Third, the labeled sentences are annotated by the annotators to produce a phrase graph containing all the annotations. Finally, the graph building stage deploys term normalization and purpose classification to transform the phrase graph into a PoliGraph.

3 PoliGraph-er Implementation

We present PoliGraph-er, the NLP-based system that we implement to generate PoliGraph from the text of a policy. Figure 4 gives an overview of its implementation.

3.1 NLP on Structured Documents

HTML Preprocessing. Policies are usually published online as structured documents, mainly in HTML format, while NLP models expect plain text input. Simply stripping HTML tags, such as headings and lists, would result in a loss of semantics. As the first step, PoliGraph-er preprocesses each HTML document to a simplified document tree which preserves three important document structures: *heading*, *list item*, and general *text*. The document tree helps to generate complete sentences as input for NLP. Please see Appendix A.1 in [19] for details.

NLP Pipeline. PoliGraph-er is built based on the spaCy library [20] and its RoBERTa-based NLP pipeline [21, 22]. The NLP pipeline labels text with linguistic labels originating from English linguistic features, including word lemmas, part-of-speech, sentence segmentation, and syntactic dependency trees (see the output of "NLP Pipeline" in Figure 4). These features are syntactic and thus require no domain adaptation.

To identify data types and entities in a policy, PoliGraph-er uses named entity recognition (NER), a standard NLP technique to classify noun phrases into a given set of labels. In our case, we use two labels: *DATA* for data types and *ENTITY* for entities. To train the NER model, we use a synthetic training set that combines generated sentences and real policy text prelabeled by an existing NER model and rule-based NER. Please see Appendix A.2 in [19] for details.

3.2 Annotators

In PoliGraph-er, we refer to the modules that identify relations between phrases[7] as *annotators*. The relations are stored as edges in a graph structure, which we call a *phrase graph*. The phrase graph is still an intermediate step, in which phrases referring to the same thing have not been merged.

Table 1: Overview of annotators in PoliGraph-er.

Annotator	**Example** (based on the policy in Figure 1(a))
Collection Annot.	Entity —COLLECT→ Data *e.g., We collect … personal information*
Subsumption Annot.	Hypernym —SUBSUME→ Hyponym *e.g., Device information… such as IP address…*
Purpose Annot.	Data —PURPOSE→ Purpose *e.g., We use your personal information … to: Provide the Services…*
Coreference Annot.	Reference —COREF→ Main mention *e.g., We collect … personal information: … We use this information to …*
List Annot.	Preceding sentence —SUBSUME / COLLECT→ List item *e.g.,* TEXT *We collect … following information:* LISTITEM - *Device information…* LISTITEM - *Location…*

To extract relations between phrases, annotators search for phrases matching specific syntactic patterns [23]. In contrast to prior work that hardcodes heuristics to perform the search [9], we use *dependency matching* to specify desired patterns as configurable rules [24]. For example, in the collection annotator, we set the rule `collect|gather|obtain|...:ROOT` to match the root verb "collect". Then the sub-rule `ENTITY:subj` matches the subject "we" under the verb as the entity, and another sub-rule `DATA:obj` matches the object "device information" as the data type. The annotator then adds a *COLLECT* edge between them in the phrase graph.

By dividing linguistic analysis tasks into five annotators, each of them focuses on a specific set of patterns. Table 1

[7] We use "phrase" to refer to verbatim words and phrases in the policy text. We use "term", which appears in previous sections as well, to refer to the normalized forms (see Section 3.3) of phrases that appear in PoliGraphs.

Table 2: Syntactic patterns used by the collection annotator.

Root Verbs *(Examples: ENTITY $\xrightarrow{COLLECT}$ DATA)*	**Syntatic Patterns**
share, trade, exchange, disclose *(We share your device IDs with Google.)*	ENTITY:nsubj DATA:dobj with,ENTITY:pobj
collect, gather, obtain, get, receive, solicit, acquire, request *(Google may collect your device IDs.)*	ENTITY:nsubj DATA:dobj
provide, supply *(We provide Google with your device IDs.)*	ENTITY:nsubj ENTITY:dobj with,DATA:pobj
provide, supply, release, disclose, transfer, transmit, sell, rent, lease, give, pass, divulge, submit *(We may transmit device IDs to Google.)*	ENTITY:nsubj DATA:dobj to,ENTITY:pobj
use, keep, access, analyze, process, store, save, hold, log, utilize, record, retain, preserve, need, maintain *(Google may use your device IDs.)*	ENTITY:nsubj DATA:dobj
have, get, gain (access to) *(Google has access to your device IDs.)*	ENTITY:nsubj access,to,DATA:pobj
make (use of) *(Google makes use of device IDs.)*	ENTITY:nsubj use:dobj of,DATA:pobj

outlines the patterns and relations that each annotator tries to identify. Note that some edge types (*COREF* and *PURPOSE*) exist only in the phrase graph and will be converted in the final PoliGraph. We discuss each annotator as follows.

Collection Annotator. The collection annotator finds affirmative sentences that disclose data collection, use or sharing, extracts entities and data types, and adds *COLLECT* edges from entities to data types in these sentences. The annotator matches around 40 verbs and 20 sets of syntactic patterns. Table 2 lists some of the patterns in the active voice. For clarity, we do not list patterns in the passive voice (*e.g.,* "this information is shared with...") and composite patterns (*e.g.,* "allow us to collect..."), but they are all handled by the annotator. We gather these patterns from actual policies in the dataset which we use to evaluate PoliGraph (see Section 5).

The collection annotator only labels *COLLECT* edges for affirmative statements. To distinguish affirmative sentences from negative and interrogative ones, it checks the existence of negative modifiers (*e.g., not*, *never*) and interrogative words in the dependency tree. While negative statements are by default excluded, we extend PoliGraph in Section 5.3 to use the information to analyze negative statements.

Subsumption Annotator. The annotator identifies subsumption relations between phrases and adds *SUBSUME* edges from a hypernym to its hyponyms. It matches 11 syntactic patterns of subsumption as shown in Table 3. These extend the patterns used in prior work [5, 9, 10].

Purpose Annotator. The annotator identifies phrases that describe purposes of data collection in three forms: (1) *in order to* ⟨*verb*⟩ ...; (2) *to* ⟨*verb*⟩ ...; (3) *for ... purpose(s)*. It links such purpose phrases to corresponding data types with *PURPOSE* edges, which are not part of PoliGraph and will be converted into $Purposes(\cdot)$ attributes on the corresponding *COLLECT* edges in PoliGraph. For example, in the sentence "We use this information *to provide ads*", the purpose phrase "to provide ads" is linked to the data type "this information".

Table 3: Syntactic patterns used by the subsumption annotator.

Phrases	**Sentences**
X such as Y_1, Y_2...	X includes Y_1, Y_2...
such X as Y_1, Y_2...	X includes but is not limited to Y_1, Y_2...
X, for example, Y_1, Y_2...	
X, e.g. / i.e. Y_1, Y_2...	
X, which includes Y_1, Y_2...	
X including / like Y_1, Y_2...	
X, especially / particularly, Y_1, Y_2...	
X, including but not limited to, Y_1, Y_2...	
Y_1, Y_2... (collectively X)	

X = hypernym phrase; Y_1, Y_2... = hyponym phrases.

Coreference Annotator. The annotator resolves pronouns (*e.g.,* "it", "they", *etc.*) or phrases modified by demonstrative determiners (*e.g.,* "this", "those", *etc.*) to the phrases which they refer to. This task, known as *coreference resolution*, is a non-trivial NLP task. Prior work [5, 9, 10] could not handle coreferences properly, which has resulted in a loss of semantics and misinterpretation of many collection statements.

We find that existing coreference resolution models [25, 26] cannot handle non-personal references well, whereas they are commonly found in policies. To address the issue, we design a heuristic-based coreference annotator that handles common forms of coreferences in policies. First, for a phrase starting with a determiner "this", "that", "these", "those" or "such" (*e.g.,* "these providers"), the annotator looks backward for the nearest phrase with the same root word (*e.g.,* "ad providers") in the same or previous sentence. Specifically, if the root word is "data" or "information" (*e.g.,* "this information"), the annotator looks backward for the nearest data type labeled by NER as the referent. Second, for a pronoun like "it", "this", "they", or "these", the annotator tries to infer whether the pronoun refers to a data type or an entity based on existing *SUBSUME* edges, and looks backward for the nearest data type or entity. The annotator links coreference phrases to the referred phrases with *COREF* edges, which are only used in phrase graphs to resolve coreferences.

We evaluate our method on 200 coreferences from our test set (see Section 4). 168 are resolved correctly. Four coreferences are partially resolved because they refer to multiple phrases while the annotator supports only one referent for each phrase. The other 28 are not resolved or are resolved wrongly. This yields 84-86% accuracy, which suggests that it outperforms general-purpose coreference models.[8]

List Annotator. This special annotator uses the document tree to discover relations between list items and their preceding sentence. First, if a noun phrase modified by "following" or "below" (*e.g.,* "the following information") precedes list items, it adds *SUBSUME* edges from the phrase to list items. Second, it propagates relations between the preceding sentence of a list and *any* list item to *all* list items in case other annotators fail to label them.

[8] For example, Coreferee [26] reported 82-83% accuracy on general corpora. However, as it does not treat data types as named entities, it cannot resolve coreferences of data types.

3.3 From Phrase Graph to PoliGraph

The final step of PoliGraph-er is to build a PoliGraph from a phrase graph. This involves merging phrases (data types and entities) with the same meaning to one node and converting purpose phrases to edge attributes.

Normalizing Data Types and Entities. PoliGraph-er starts by mapping data types and entities in the phrase graph to their normalized forms. For example, "contact details" and "contact data" are synonyms to the normalized term "contact information" which we want to keep in the PoliGraph.

For data types and entities in our global ontologies (see Section 2.2), we consider them as standard terms and write regular expressions to capture their synonyms. For example, the regular expression `contact\b.*\b(information|data|detail|method)` matches synonyms of "contact information". PoliGraph-er maps these synonyms to "contact information", which aligns with the term in the global data ontology, as the normalized form. We also programmatically create regular expressions for variants of company names from public datasets. If a phrase is not a standard term and thus does not match any regular expressions, PoliGraph-er simply strips stop words and takes the lemmatized form of the phrase as the normalized form. For example, "your vehicle records" is normalized to "vehicle record". This is usually enough to capture variants of the same term caused by word inflections.

For coreferences, as the annotator has linked each coreference phrase to what it refers to, PoliGraph-er follows the *COREF* edge to find the referred phrase and use the same normalized form. For example, in Figure 1(a), "this information" would be normalized to "geolocation", same as the phrase "location" that it refers to. Please see Appendix A.3 in [19] for details on the phrase normalization.

Unspecified Data and Unspecified Third Party. As a special case of phrase normalization, policies often use blanket terms like "information" and "third party" without further details. For example, in the sentence "We collect information to provide services", the word "information" can be interpreted as unspecified (or all possible) data types. We find it more appropriate to treat such blanket terms specially than assuming them to have consistent meaning across the text. PoliGraph-er uses two special nodes, "unspecified data" for data types and "unspecified third party" for entities, as the normalized forms of such blanket terms in PoliGraphs. Please see Appendix A.3 in [19] for details.

Classifying Purpose Phrases. The purpose annotator identifies purpose phrases. To allow automated analysis of purpose, we coarsely group purpose phrases into five categories: *services*, *security*, *legal*, *advertising*, and *analytics*. These categories are derived from the business and commercial purposes defined in the CCPA [2]. As in prior work [6], we distinguish between *core* (*i.e.,* services, security, and legal) and *non-core* (*i.e.,* advertising and analytics) purposes of data collection.

We fine-tune a sentence transformer model [27] to classify purpose phrases into these categories. For example, the phrase "to provide features" is classified as *services*, whereas the phrase "for advertising purposes" is classified as *advertising*. Note that a purpose phrase can be classified into multiple labels if it mentions more than one purpose. To train the model, we manually annotate a dataset of 200 phrases. We use SetFit [28] to enable few-shot fine-tuning with this small dataset. This can be done because the underlying transformer model is already trained on a large corpus to gain language knowledge, and SetFit takes advantage of contrastive learning to learn the differences between classes effectively. The performance of purpose classification is reported in Section 4.

Building PoliGraph. Finally, PoliGraph-er builds the PoliGraph from the phrase graph by merging phrases with the same normalized form into one node, keeping *COLLECT* and *SUBSUME* edges, and inferring the $Purposes(\cdot)$ attributes from *PURPOSE* edges in the phrase graph.

Figure 5 shows an example of a PoliGraph generated from a simple policy [29] for demonstration purposes. Due to a lack of space, we show a subset of data types. The complete version can be found in Figure 10 in Appendix A.4 in [19]. A typical PoliGraph from the policies that we have analyzed (see Section 5) can contain up to hundreds of nodes and edges. It is common to see vague phrases like "statistical user data" that are not further clarified, and misleading definitions like claiming anonymized information to include data types that are likely personal and identifiable. However, it is important that PoliGraph does capture data collection, along with its purposes, and subsumptions for further analysis.

4 PoliGraph-er Evaluation

In this section, we evaluate PoliGraph-er's performance in analyzing policies. At the beginning of each subsection, we state the research question (RQ) addressed therein, our approach, and a preview of the results.

The PoliCheck Dataset. Throughout this section and Section 5, we use the public dataset provided by PoliCheck [10]. We choose this dataset because it is among the largest public datasets for policies. The dataset consists of policies of 13,796 Android apps on Google Play Store. The number of policies is large enough to necessitate automated analysis. Furthermore, it also comes with the apps' network traffic data that facilitate flow-to-policy consistency analysis in Section 5.4. We write a crawler script based on the Playwright library [30] to download the policy text from each URL. We obtain the most recent version of the policies from March 2023. After excluding non-English, invalid, and duplicate webpages, we obtain 6,084 unique policies used by 13,626 apps.

Test Set. Out of the full PoliCheck dataset, we randomly select a set of 200 policies, and we read and annotate them in order to build ground truth for various evaluation tasks. This test set has no overlap with the data used to generate the NER corpus or to train the purpose classifier (see Section 3).

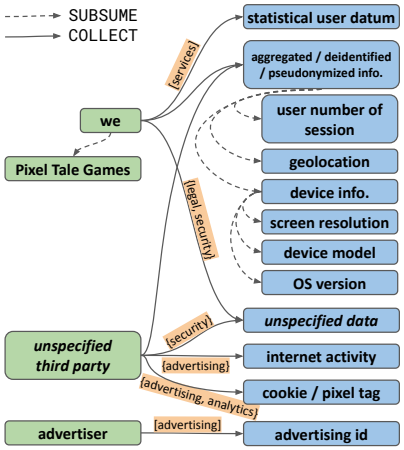


Figure 5: The PoliGraph generated from "Puzzle 100 Doors" app's policy [29].

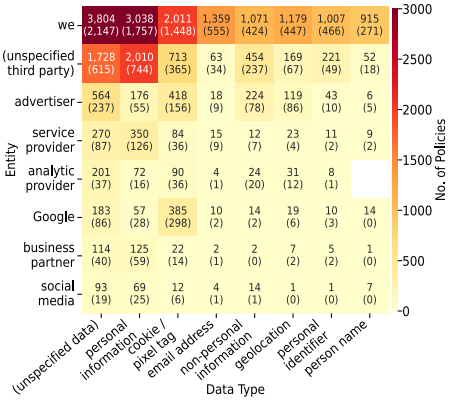


Figure 6: Statistics of common *COLLECT* edges. The numbers of edges that have *Purposes*(·) attributes are shown in parentheses.

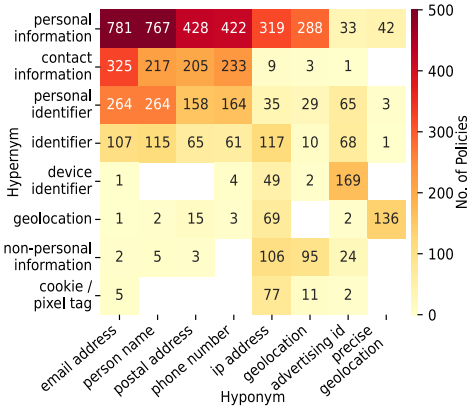


Figure 7: Statistics of common *SUBSUME* edges between data types.

4.1 PoliGraph Generation

RQ1. How accurate is PoliGraph-er *in generating* PoliGraph*s from policies w.r.t. the definitions in Section 2.1?*

To answer this question, we present the statistics of PoliGraph's *COLLECT* and *SUBSUME* edges, as well as manual validation of the precision of these edges. The tool successfully generates PoliGraphs for 5,255 policies. The remaining policies cannot be processed because they either do not claim to collect data, or use irregular or unsupported HTML tags that cannot be correctly parsed.

4.1.1 Characterization of PoliGraph Edges

First, we characterize the *COLLECT* and *SUBSUME* edges in the PoliGraphs generated by PoliGraph-er.

***COLLECT* Edges.** PoliGraph-er extracts 103,185 *COLLECT* edges from 100,565 sentences in total. Among them, 34,052 edges have *Purposes*(·) attributes from 38,994 purpose phrases. Figure 6 shows the common *COLLECT* edges found in the dataset. Generic terms, such as "personal information" and "personal identifier", are commonly used to express data types in policies. Some specific terms, such as "cookie / pixel tag", "email address", and "ip address" are also found in many policies. Furthermore, we find that policies disclose data collection by first-party (*i.e.,* "we") more frequently than by third-party entities. Major third-party entity categories are "advertiser" and "analytic provider". Google, as the platform, is also frequently mentioned in the policies.

***SUBSUME* Edges.** PoliGraph-er extracts 52,007 *SUBSUME* edges from 20,959 sentences in the dataset. Figure 7 shows common *SUBSUME* edges that connect data type nodes. "Personal information", "contact information" and "personal identifier" are the most frequently used generic terms to represent data types. Notably, we find that many policies declare the collected data as "non-personal information": this conflicts with our CCPA-based global data ontology. We will discuss the issue of misleading definitions in Section 5.2.

"Unspecified". The nodes "unspecified data" and "unspecified third party" (see Section 3.3) are found in 72.0% (3,785) of PoliGraphs. This is because many policies discuss data collection, sharing, and use in separate sections. When they discuss sharing, precise data types are often omitted. When they discuss purposes of use, both data types and entities can be unspecified terms. For example, KAYAK's policy states: "To protect rights and property, we may disclose your information to third parties" [4]. Without further details on "information" and "third parties", the statement is captured in the PoliGraph as *unspecified third party* $\xrightarrow{\text{COLLECT}}$ *unspecified data* with *security* as the purpose.

4.1.2 Manual Validation of PoliGraphs

We manually evaluate whether PoliGraph-er extracts the correct edges. To evaluate the *precision* of PoliGraph edges, we sample five edges from each of 100 randomly selected PoliGraphs in the dataset and read the corresponding policy text to validate whether each edge is correctly extracted from the text. To help with this evaluation, PoliGraph-er stores the sentences from which each edge is generated. We find that the precision for *COLLECT* edges is 90.4%, and the precision for *SUBSUME* edges is 87.7%.[9]

In theory, false positive edges can propagate to more incorrect inferences. For example, a false *advertiser* $\xrightarrow{\text{COLLECT}}$ *personal information* edge would lead to wrong inferences of *advertiser* collecting all data types subsumed by *personal information*. However, we find that such cases are rare, and false positives are often caused by recognizing irrelevant phrases as data types or entities[10], which is less of an issue if we scope the analysis to a subset of common data types or entities (as we show in Section 4.2). Most false-positive edges come from NLP errors including: (1) NER recognizing irrelevant phrases

[9] We do not evaluate recall as it turned out to be difficult for humans to label edges without biases for all data types and entities. We will report recalls in Section 4.2 where we consider a subset of common data types.

[10] For example, in the sentence "the app may use third party code", PoliGraph-er mistakes "third party code" as a data type.

as data types or entities, and (2) mistaking some interrogative or negative sentences as affirmative statements.

We also evaluate POLIGRAPH-ER's purpose classification model. We randomly sample five purposes phrases identified by the purpose classifier from each policy in the test set, manually assign labels to the phrases, and compare them to the ones labelled by the purpose classification model. Overall, the *macro-averaged precision* and *recall* are 91.0% and 94.8%, respectively, for this multi-label multi-class classification task.

4.2 Comparison to Prior Policy Analyzers

RQ2. How well does POLIGRAPH-ER *analyze collection statements compared to prior state-of-the-art policy analyzers?*

To answer this question, we use POLIGRAPHs to infer collection relations, namely $collect(n,d)$ indicating that an entity n may collect a data type d. We obtain tuples of collection statements from prior state-of-the-art, namely PolicyLint [9]. We compare both results to manually labeled ground truth.

Methodology. Since PolicyLint extracts tuples (see Figure 1(b)), we convert pairs of $collect(n,d)$ relations in POLIGRAPHs into PolicyLint tuples $\langle n, collect, d\rangle$. Still, we cannot compare data types and entities from the two tools directly because they normalize phrases in different ways. To work around the issue, we select a subset of terms to compare. For data types, we only consider the following precise data types in PolicyLint, since they are comparable to the same data types extracted by POLIGRAPH-ER: "mac address", "router ssid", "android id", "gsf id", "sim serial number", "serial number", "imei", "advertising identifier", "email address", "phone number", "person name", and "geographical location".[11] For entities, we only distinguish between the first party and third party, *i.e.,* all tuples are converted to either $\langle$*we, collect, data type*$\rangle$ or $\langle$*third party, collect, data type*$\rangle$; "unspecified third party" in POLIGRAPH is considered a third party.

POLIGRAPH-ER finds 13,529 tuples in the entire dataset. PolicyLint finds 6,410 tuples. To evaluate the precision and recall of both tools, we manually extract the same tuples (collection relations) from our test set to establish the ground truth. To do this, we first use coarse regular expressions to match all possible mentions of the 12 data types in policies with a likely high chance of false positives. Then two of our authors read the text to determine if each data type is collected by the first party or any third party to create the tuples.

Precision. POLIGRAPH-ER achieves 96.9% precision, and PolicyLint achieves 91.8% precision. As previously explained, NLP errors are the main reason of wrong collection relations. We improve the precision by using recent NLP models. The precision of POLIGRAPH-ER is higher than reported in Section 4.1.2 because here we only consider a subset of data types and thus many falsely labeled data types are excluded. Also note that both tools show lower precision for third-party tuples (see Table 4) because some policies use company names rather than "we" to refer to the first party, and both tools can mistake the company names as third parties in this case.

Table 4: Manual validation and ablation studies results

	# tuples	prec. (1st/3rd party)	recall (1st/3rd party)
Manual Validation			
Ground Truth	878	-	-
POLIGRAPH-ER	640	96.9% (99.8% / 91.9%)	70.6% (64.0% / 87.5%)
PolicyLint	291	91.8% (93.0% / 82.4%)	30.4% (37.9% / 11.3%)
Ablation Studies			
no-subsumption-annot.	345	96.5% (99.6% / 89.4%)	37.9% (38.1% / 37.5%)
no-coreference-annot.	616	96.9% (100% / 91.4%)	68.0% (62.5% / 81.9%)
no-list-annotator	614	97.1% (100% / 92.0%)	67.9% (61.6% / 83.9%)
per-sentence-extraction	471	97.0% (100% / 89.8%)	52.1% (53.0% / 49.6%)
per-section-extraction	573	97.0% (99.7% / 91.5%)	63.3% (61.0% / 69.4%)

Recall. POLIGRAPH-ER achieves 70.6% recall, compared to PolicyLint's 30.4% recall. As we will explain later in the ablation studies, the graph structure and improved NLP techniques both contribute to the higher recall. Despite the improvement, the recall of our tool is still limited, mainly by its linguistic analysis approach. First, some policies use lists or tables, which does not contain complete sentences for analysis. Second, our annotators cannot capture all forms of collection statements and can miss data types and entities in long or convoluted sentences.

Given the high precision and improved, but imperfect, recall, we recommend to interpret what POLIGRAPH-ER captures as a lower bound of the actual collection statements. Appendix D in [19] discusses the factors that impact the performance of POLIGRAPH-ER in more detail.

4.3 Ablation Studies

RQ3. POLIGRAPH *consists of many components. Where does the performance improvement come from?*

To answer this question, we conduct ablation studies to understand how each component and design decision contributes to POLIGRAPH-ER's performance improvements. We modify POLIGRAPH into the following experimental configurations, and we summarize the evaluation results in Table 4.

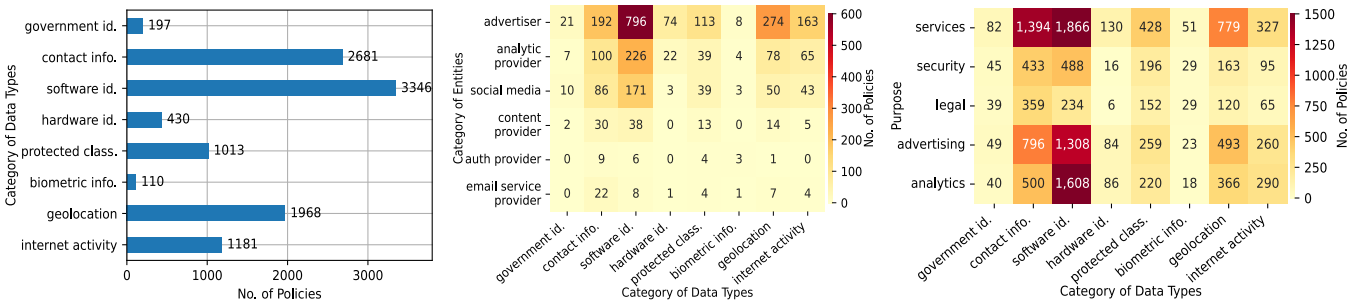
Removing Components. In the *no-subsumption-annotator*, *no-coreference-annotator*, *no-list-annotator* configurations, we disable one component at a time and assess the effect.

As shown in Table 4, disabling the subsumption annotator reduces recall from 70.6% to 37.9%. The reason is that the precise data types that we evaluate are often subsumed by generic terms. They have to be linked by *SUBSUME* edges to allow inferences of collection relations. Each of the coreference and list annotators contributes about 3% to recall.

Limiting Extraction Area. In the *per-sentence-extraction* configuration, we modify POLIGRAPH-ER to behave like PolicyLint and extract tuples within sentence boundaries[12]. This is done by filtering out $collect(n,d)$ relations in POLIGRAPHs where the entity and data type comes from different sentences.

[11] Note that we map "coarse geolocation", "precise geolocation", and "geolocation" in POLIGRAPH all to "geographical location" in the tuple because PolicyLint does not distinguish between them.

[12] The *per-sentence-extraction* is different from removing *SUBSUME* edges because PolicyLint ideally is still able to find data types and entities that are subsumed by generic terms within the same sentence.



(a) Collection of Data Types (b) Third-party Entities of Data Collection (c) Purposes of Data Collection

Figure 8: Policy Summarization — Statistics of policies that disclose **(a)** the collection of eight categories of data types; **(b)** data collection by third-party entities (per data type category); and **(c)** the purposes for the collection (per data type category). For example, 3,348 policies claim to collect data types under the "software identifier" category; 794 policies disclose that data types in this category are collected by "advertiser" as the third-party entity. And 1,847 policies disclose the collection of this category for the "services" purpose.

Similarly, *per-section-extraction* only considers collection relations within the boundaries of sections[13]. These configurations help assess the improvement by introducing the graph structure to infer relations across sentences and sections.

Table 4 shows the results. The *per-sentence-extraction* configuration has only 52.1% recall compared to PoliGraph-er's 70.6%. On one hand, the sophisticated NLP methodology still improves performance over PolicyLint. On the other hand, the graph structure is necessary to infer 18.5% of all relations. The graph is more effective in identifying third-party data collection—it allows us to find 37.9% of third-party tuples, because the disclosure of third-party sharing often uses broader terms (*e.g.,* "*Anonymous information* may be shared with analytic providers") and the exact data types (or entities) must be inferred through PoliGraph. The *per-section-extraction* configuration, by connecting data types and entities within a longer textual region, achieves 63.3% recall but still falls behind the full version by missing 7.3% of relations. As explained in Section 1, the typical structure of policies that discuss data collection, use and sharing in separate sections makes it necessary to find exact data types and entities in different sections. Therefore, we argue that PoliGraph enables a much better coverage by connecting information disclosed in different sentences and sections to infer more collection relations in a policy.

5 PoliGraph Applications

In this section, we present two novel applications enabled by PoliGraph. Section 5.1 presents policies summarization, which provides inferences on the common patterns across different policies. Section 5.2 looks into how the same or similar terms are defined across different policies. In addition, we show that PoliGraph can improve two applications that have been explored by prior work. Section 5.3 extends PoliGraph to identify contradicted statements. Section 5.4 applies PoliGraph to check the consistency between the data flows of a mobile app and its policy.

[13]As explained in Section 3.1, PoliGraph-er keeps headings from HTML, which is considered as the boundaries of sections here.

5.1 Policies Summarization

We use PoliGraph to summarize *all* policies in our dataset and reveal common patterns among them. Specifically, we aim to identify: *(1) how common each category of data types is collected; (2) what kind of entities collect these data types;* and *(3) the purposes for which these data types are used.* As data types and entities captured by PoliGraphs are unorganized and differ across policies (see Section 4.1.1), we use our global ontologies to categorize data types and entities in a canonical manner.

Data Types. In this analysis, we use the eight parent nodes of the leaf nodes in the data ontology shown in Figure 2 to group the data types into eight categories[14]: "government identifier", "contact information", "software identifier", "hardware identifier", "protected classification", "biometric information", "geolocation", and "internet activity". Figure 8a shows the numbers of policies that disclose the collection of data types in each category. Overall, 77.9% (4,093) of policies disclose the collection of at least one of these data categories.

Finding 1. *The most frequently collected data category is "software identifier", which mostly originates from "cookie" as the specific data type being collected.* 63.7% (3,346) of policies disclose the collection of "software identifier". Among the specific data types, "cookie / pixel tag" is the most common and found in 80.3% (2,688 / 3,346) of these policies. On the other hand, identifiers specific to mobile apps, mainly "advertising ID" and "Android ID", are found in only 26.0% (870) and 2.8% (95) of these policies, respectively. Many developers simply write one policy for various products, including mobile apps and web-based services. Furthermore, some developers seem to use "cookie" as a generic term for all kinds of device identifiers for tracking.

Third-Party Entities. We use the six parent nodes of the leaf nodes in the entity ontology shown in Figure 3 to group entities into six categories[14]: "advertiser", "analytic

[14]"Unspecified data", "unspecified third party" and other data types and entities that are not part of the global ontologies (see Section 2.2) are excluded from the analysis of policies summarization.

provider", "social media", "content provider", "auth provider", and "email service provider". Figure 8b reports how each data type category is disclosed to be shared with or collected by these third-party entities (*i.e., collect*(n,d) relations).

Finding 2. *"Software identifier" is frequently shared with advertisers. Third-party sharing of other data categories (e.g., "geolocation", "protected classification", and "internet activity") is also non-negligible.* We find that 23.8% (796 / 3,348) of policies that disclose to collect "software identifier" involve sharing with advertisers. Analytic providers and social media are other major third parties with whom apps share data. 620 policies share data in other categories (*e.g.,* "geolocation", "internet activity", and even "protected classification") with third parties. As data in these categories may be sensitive, it is doubtful whether the sharing of them is appropriate.

Finding 3. *Many policies disclose data sharing using generic terms, e.g., "personal information". This leads to the inference that the app may share all data types that a generic term subsumes.* This often happens when policies disclose data collection, sharing, and use separately. For example, Figure 1 shows that "personal information" is shared with entities such as "social networking services". This may be alarming to users since "personal information" subsumes sensitive data types, such as "location" and "IP address". The use of generic terms reduces transparency. Users are left wondering which, if not all, "personal information" is shared. We find that 710 policies declare third-party collection or sharing using generic terms that subsume data types in multiple categories.

Purposes. Figure 8c reports the statistics of policies that disclose purposes of data collection, as discussed in Section 3.3, per data type category.

Finding 4. *56.9% (2,990) policies disclose that their apps collect data for non-core purposes. In over 80% of them (2,398), the main non-core purpose is advertising.* We find that, while "software identifier" remains the most common data type category used for non-core purposes, the potential use of other data types for non-core purposes is concerning. For instance, the collection of "geolocation", "protected classification", and "internet activity" for non-core purposes is declared in about 10% of policies. The CCPA [2] defines "government identifiers", "precise geolocation", and certain protected classifications as *sensitive personal information*—the law limits the usage of these data (*e.g.,* users have the right to limit the use of such personal information for non-core purposes).

5.2 Correct Definitions of Terms

The second novel application enabled by PoliGraph is assessing the correctness of definitions of terms. Besides summarizing policies on their own right, we can check whether a policy defines terms in ways that are consistent with external knowledge as captured by global ontologies. This is necessary because policies often provide their own definitions of terms. This is not a problem if the definitions align with external knowledge (*e.g.,* privacy laws), but it may be *misleading* if they do not agree. For example, some policies define "geolocation" as "non-personal information". In this section, we check whether the definitions of data type terms in individual policies (as captured by their PoliGraphs' local ontologies) align with our CCPA-based global data ontology (see Figure 2). Overall, such different definitions are found in 25.5% (1,339 / 5,255) policies in our dataset, as listed in Table 5.

Table 5: Examples of different definitions found in PoliGraphs with respect to the global data ontology. For example, "geolocation" is defined as "non-personal information" in 123 policies.

Hypernym	**Hyponym (# Policies)**
non-personal info.	ip address (126), geolocation (123), device identifier (108), gender (76), application installed (72), age (70), identifier (46), internet activity (44), device information (38), coarse geolocation (35) ...
aggregate/deidentified/ pseudonymized info.	ip address (122), device identifier (89), geolocation (78), browsing / search history (16) ...
internet activity	ip address (151), device identifier (107), geolocation (40), advertising id (13), cookie / pixel tag (10) ...
geolocation	ip address (76), postal address (15), router ssid (10) ...
personal identifier	advertising id (74), cookie / pixel tag (49), device identifier (39), geolocation (35), date of birth (27), gender (23) ...

Finding 5. *Many policies define data types that they collect to be "non-personal", "aggregated", "deidentified", or "pseudonymized". However, this can be inconsistent with the definitions in the CCPA.* Indeed, in the CCPA, "deidentified information" is defined as information that "cannot reasonably identify, relate to, describe ... to a particular consumer". Although entities technically can deidentify personal information, some of the data types we observe in Table 5, notably "geolocation", "gender", "age", and "date of birth", are generally considered personal information by the public and according to the CCPA. Declaring these data types as *non-personal* or *deidentified* can be misleading. For example, Paleblue declares in its policy [31] that "Paleblue may also invite you to share *non-personal information* about yourself which may include... (1) your age or date of birth; (2) your gender...".

Finding 6. *Many policies use non-standard terms. They can have broad or varied definitions across different policies.* For example, it is not surprising that the definition of "profile information" is application-specific. One policy from the Manager Readme app [32] defines "profile information" to include "name" and "location", while another policy from Armor Game Inc. [33] defines the term to include "gender" and "birthday". In these cases, the use of non-standard terms is acceptable as the policies clearly explain what they mean by the terms. However, we also find many policies that do not clearly define their non-standard terms. Particularly, while "profile information" is found in 178 policies, subsumption relationships are found in only 17 of them in their corresponding PoliGraphs. Table 6 presents examples of non-standard terms and their possible definitions found in the policies.

Table 6: Examples of non-standard terms found in policies. For example, "technical information" is used in 311 policies but its detailed definition is only found in 126 policies.

Term (# Policies)	Possible definitions found in policies
technical info. (311)	*From 126 policies:* advertising id, age, android id, browsing / search history, cookie / pixel tag, device identifier, email address, geolocation, imei, ip address, mac address ...
profile info. (178)	*From 17 policies:* age, contact information, date of birth, email address, gender, geolocation, person name, phone number ...
demographic info. (315)	*From 112 policies:* age, browsing / search history, date of birth, email address, gender, geolocation, ip address, postal address, precise geolocation, race / ethnicity, router ssid ...
log data (81)	*From 52 policies:* advertising id, android id, cookie / pixel tag, coarse geolocation, cookie / pixel tag, email address, geolocation, imei, ip address, mac address, person name ...

5.3 Contradiction Analysis

In this section, we apply POLIGRAPH-ER to analyze contradictions within a policy. To that end, we extend POLIGRAPH to also analyze negative collection statements (*e.g.,*"We do not collect personal data") ignored in previous sections (which only analyzed affirmative statements). We also propose extensions to the main POLIGRAPH framework so as to capture additional contexts that are crucial to interpret contradictions.

Prior work, namely PolicyLint [9], identifies contradictions by detecting affirmative and negative sentences that mention the same or conflicting entities and data types. By manually checking all the 86 contradictions identified by PolicyLint in our test set, we found 79 of them turn out to be false alarms. This is because the analysis ignores many contexts surrounding the "contradicted" statements, as explained below.

- *Different Purposes:* PolicyLint considers "we collect PII to provide the service" and "we do not collect PII for advertising purposes" as contradicting statements, despite them discussing data collection for different purposes, *i.e.,* services *vs.* advertising.
- *Different Data Subjects:* PolicyLint considers "we collect PII" and "we do not collect PII from minors" as contradicting. A human reader would recognize that the sentences discuss different data subjects, *i.e.,* general user *vs.* child.[15]
- *Different Actions:* PolicyLint considers "... share PII with third parties" and "... do not sell PII to any third party" as contradictions. A human reader would recognize that they refer to different actions, *i.e.,* share *vs.* sell.
- *Contradictions According to Global Ontologies:* In addition, some policies do not literally contradict themselves, but the data types and entities in the affirmative and negative statements overlap according to PolicyLint's ontologies. For example, in the policy of Horizone Media [34], PolicyLint reports "we use anonymous identifiers" and "we do not collect personal information" as a contradiction. The policy does not define "anonymous identifiers" as "personal information", but PolicyLint views all "identifiers" as "personal information" in its data ontology. In this case, the policy can be considered as misleading but does not directly contradict itself. Note that we already discuss misleading definitions in Section 5.2.

5.3.1 Framework Extensions

The main POLIGRAPH framework, described in Section 2, introduced data types, entities and purposes. However, it only deals with affirmative, not negative, statements. In this section, we extend POLIGRAPH to also analyze negative statements, and we show how to deal with contradicted statements in fine-grained contexts. To that end, we add a new type of negative edge (*NOT_COLLECT*), actions as edge subtypes, and the notion of data subject, as described next.[16]

Negative Collection Statements. The collection annotator (see Section 3) identifies negative sentences and by default excludes them. In this section, we modify the annotator to also account for negative collection statements and represent them as *NOT_COLLECT* edges. For example, "we do not collect personal information" will be represented with edge *we* $\xrightarrow{\text{NOT\_COLLECT}}$ *personal information* in POLIGRAPH. Similarly to its positive counterpart, a *NOT_COLLECT* edge can have $Purposes(\cdot)$ attributes.

Refining Actions. We further consider five subtypes of *COLLECT* (and *NOT_COLLECT*) edges that represent different data actions: $\{collect, be\_shared, be\_sold, use, store\}$. We denote an action-sensitive *COLLECT* edge as $n \xrightarrow{\text{COLLECT}\ [a_{pos}]} d$, and an action-sensitive *NOT_COLLECT* edge as $n \xrightarrow{\text{NOT\_COLLECT}\ [a_{neg}]} d$, where a_{pos} and a_{neg} are one of the 5 subtypes of data actions.

We extend the collection annotator to map verbs to these subtypes accordingly. For example, "... do not sell personal information to advertisers" is represented as edge *advertisers* $\xrightarrow{\text{NOT\_COLLECT}\ [be\_sold]}$ *personal information* in POLIGRAPH. Table 9 in Appendix B.1 in [19] provides the list of verbs and corresponding actions, which can be extended if so desired.

Data Subjects. We extend data type nodes to include subjects, *i.e.,* the group of people to which the data type pertains, following the terminology of Contextual Integrity [35]. We denote a subject-sensitive data type node as a pair (d, s), where d is the data type, and s is the subject of the collected data.

We add a new subject annotator in POLIGRAPH-ER to identify subjects of data types. Currently, we implement it to identify a commonly-seen data subject: children. In the current

[15] Beyond semantics, another source of error for PolicyLint was its implementation. In order to find and skip text about children policy, which was considered outside the framework, it performed string matching using hardcoded regular expressions, which often failed due to sentence variability.

[16] The extensions are presented in this section, as opposed to as part of the main framework, for several reasons. First, they are specific to contradiction analysis, and motivated by the limitations of prior work in taking into account fine-grained contexts in that analysis. Second, as shown in Section 5.3.3, although effective in refining contexts for contradiction analysis, these extensions do not address *all* aspects of fine-grained contexts: *e.g.,* one can define additional types of actions, data subjects, and other contexts. Third, these extensions do not affect the validity of previous results in Sections 4 and 5, which were based only on affirmative, and ignored negative, statements.

implementation, we define subject $s \in \{child,\ general\ user\}$, and we represent the statement "we don't collect personal information from children" as $we \xrightarrow{\text{NOT\_COLLECT}\ [collect]} (personal\ information,\ child)$ in PoliGraph. We note, however, that the modular implementation of PoliGraph-er allows for the set of subjects to be extended to include additional subjects.

5.3.2 Contradiction Analysis

To identify potential contradictions, we assess whether a positive edge and a negative edge in a PoliGraph involve conflicting data types, subjects, entities, actions, and purposes. A pair of edges may contradict if all these parameters conflict; otherwise, the two statements address different aspects of data collection, as outlined earlier, thus do not contradict.

Definition 5.1. ***Conflicting Edges.*** A positive edge $e_{pos} = n_{pos} \xrightarrow{\text{COLLECT}\ [a_{pos}]} (d_{pos}, s_{pos})$ and a negative edge $e_{neg} = n_{neg} \xrightarrow{\text{NOT\_COLLECT}\ [a_{neg}]} (d_{neg}, s_{neg})$ in a PoliGraph are conflicting if all the following parameters conflicts:

- ***Data types*** d_{pos} and d_{neg} conflict *iff* $d_{pos} = d_{neg}$ or $\exists d'$: $subsume(d_{pos}, d') \wedge subsume(d_{neg}, d')$.
- ***Entities*** n_{pos} and n_{neg} conflict *iff* $n_{pos} = n_{neg}$ or $\exists n'$: $subsume(n_{pos}, n') \wedge subsume(n_{neg}, n')$.
- ***Purposes*** $P_{pos} = Purposes(e_{pos})$ and $P_{neg} = Purposes(e_{neg})$ conflict *iff* (1) $P_{pos} \cap P_{neg} \neq \varnothing$, or (2) $P_{neg} = \varnothing$.
- ***Data subjects*** s_{pos} and s_{neg} conflict *iff* $s_{pos} = s_{neg}$.
- ***Actions*** a_{pos} and a_{neg} conflict *iff* $a_{pos} = a_{neg}$.

5.3.3 Results on Contradiction Analysis

Reducing False Alarms. PoliGraph with the above extensions encodes additional parameters, which allows us to reclassify many of the statements erroneously classified as candidate contradictions in PolicyLint, due to missing context, as non-contradictions. To evaluate the benefit, (1) we map contradicting tuples reported by PolicyLint to PoliGraph edges and (2) we check whether each pair of edges are conflicting as defined above. PoliGraph-er maps 2,555 PolicyLint contradictions to 1,566 pairs of edges in our dataset. Out of them, only 13.5% (211) pairs are conflicting, taking contexts into account as per Definition 5.1. The remaining 86.5% are not conflicting, due to one or more of the reasons shown in Table 7. Please see Appendix B.2 in [19] for details.

Validation. We manually verified all the 83 contradictions reported by PolicyLint in our test set. PoliGraph-er reported 68 pairs as non-conflicting and we agree with all of them. For the other 15 pairs of conflicting edges, 7 pairs of edges are considered contradictions by human readers.

Despite the additional contexts captured by our extended framework compared to PolicyLint, PoliGraph-er still has false alarms. We manually verify all 211 pairs of conflicting edges identified by PoliGraph-er, and find that only 25.1% (53) pairs are real contradictions. The most common reason (54.0%, 114 pairs) why conflicting edges may not be real contradictions is additional language nuances, not yet represented in PoliGraph. For example, the sentence "we do not collect personal data when you visit the site" does not contradict with "we collect personal data when you sign up" due to the different conditions addressed. Other contexts we identify during manual validation include data sources and consent types. Please see Appendix B.3 in [19] for details.

Table 7: Reclassification of PolicyLint contradictions

	# pairs of edges	
Invalid*	183	(11.7%)
Non-conflicting parameters	731	(46.7%)
Different purposes	*114*	*(7.3%)*
Different data subjects	*121*	*(7.7%)*
Different actions	*624*	*(39.8%)*
Contradictions according to PolicyLint's ontologies	441	(28.2%)
Conflicting edges	211	(13.5%)
Total	1,566	

* "Invalid" means that PoliGraph-er maps a negative PolicyLint tuple to a positive edge or the reverse. This is often because PolicyLint misinterprets positive or negative sentences due to NLP limitations.

Conclusion. Our extensions of PoliGraph to analyze contradictions, by taking account of more contexts, prevent many false alarms in prior work and narrow down possible contradictions to a smaller set. However, one should be aware that even the extended framework does not cover all possible contexts and nuances in human language.

5.4 Data Flow-to-Policy Consistency Analysis

In this section, we compare the statements made in a policy (extracted using PoliGraph) to the actual data collection practices (as observed in the network traffic generated). This application has been previously explored by PoliCheck for mobile apps [10] and its adaptations to other app ecosystems, *e.g.,* smart speakers [12, 13] and VR devices [6].

5.4.1 Data Flow-to-Policy Consistency

As in prior work [5, 10], we represent data collection practices observed in the network traffic as data flows. A data flow is a tuple $f = (n, d)$ where d is the data type that is sent to an entity n. Given a PoliGraph, we check whether the data flow is clearly disclosed in it as below.

Definition 5.2. ***Clear Disclosure.*** Following Definition 2.1, $G = \langle D, N; E_S, E_C; P \rangle$ is a PoliGraph. A data flow $f = (n, d)$ is *clearly disclosed* in the policy represented by G *iff* it contains the entity ($n \in N$) and the data type ($d \in D$), and $collect(n, d)$ is true in G.

Recall that PoliCheck also accepts broader terms of data types and entities as consistent but *vague* disclosure, if the broader terms subsume the specific data types in the data flows according to the global ontologies. We define vague disclosures of a data flow in a similar way as follows.

Definition 5.3. ***Vague Disclosure.*** Following Definitions 2.1 and 2.6, $G = \langle D, N; E_S, E_C; P \rangle$ is a PoliGraph, $O_d = \langle D_d, E_d \rangle$ is the global data ontology, and $O_n = \langle N_n, E_n \rangle$ is the global entity ontology. A data flow $f = (n, d)$ is *vaguely disclosed* in G according to global ontologies O_d and O_n, *iff* f is not clearly disclosed in G, and there exist a data type $d' \in D \cap D_d$ and an entity $n' \in N \cap N_n$ that satisfy: $collect(n', d')$ is true in G; $subsume(d', d)$ is true in O_d; and $subsume(n', n)$ is true in O_n.

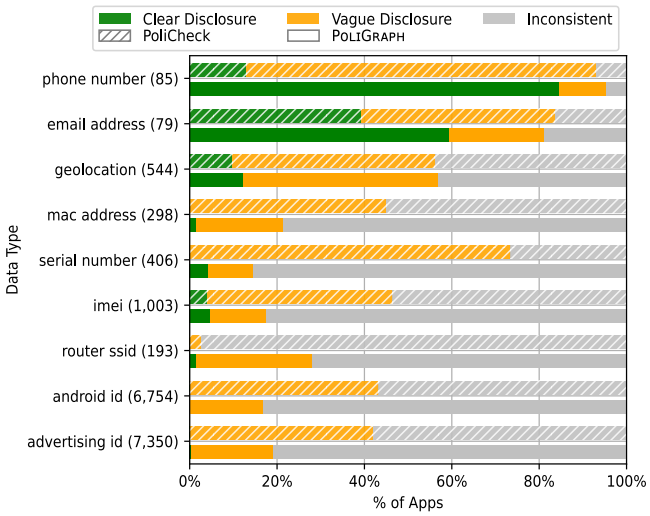


Figure 9: Flow-to-policy consistency comparison of PoliGraph *vs.* PoliCheck. The numbers in parentheses represent the number of apps that collect the data type. For example, 85 apps have data flows that collect "phone number".

Our definitions of clear and vague disclosures correspond to the same concepts in PoliCheck. Both clear and vague disclosures are considered *consistent disclosures*. Otherwise, the data flow has *inconsistent disclosure* in the policy.

5.4.2 Data Flow-to-Policy Consistency Results

Dataset. To facilitate a comparison to PoliCheck, we use the same dataset from it [10]. In addition to policy URLs for apps, this dataset also contains data flows extracted from the apps' network traffic. Different from the policy dataset in Section 4, we use the versions of policies from 2019 for a fair comparison with PoliCheck results, because the data flows were collected around February 2019. We crawl the historical versions of policies from Internet Archive [36]. In total, we have 8,757 apps with both data flows and policies available.

Figure 9 compares flow-to-policy consistency results per data type in PoliGraph *vs.* PoliCheck. An app may send one data type to multiple entities, resulting in multiple data flows per data type. In this case, we report the worst disclosure type for the app, *e.g.,* if, at least, one of the data flows of the data type is inconsistent, the disclosure type is reported as inconsistent. We present the results for nine data types—three out of 12 data types are analyzed in [10], but excluded here as only less than 10 apps exhibit the three in their data flows.

Clear disclosures. We find that PoliGraph-ER outperforms PoliCheck in terms of capturing more clear disclosures. PoliCheck underestimates the number of clear disclosures for all data types due to its limited recall (see Section 4.2). Clear disclosure of contact information is especially underestimated. 72 of 85 apps that collect "phone number", and 47 of 79 apps that collect "email address" clearly disclose the collection.

Vague disclosures. Here, PoliGraph-ER extracts fewer vague disclosures than PoliCheck. Further investigation reveals that this is because our global data ontology has a different design compared to PoliCheck's. In PoliCheck's data ontology, "personal information", a commonly seen term in policies, subsumes "device identifiers". While this would effectively increase PoliCheck's coverage, namely that the collection of all data types related to "device identifiers" found in the data flows would be categorized as vague disclosures, it is unclear whether "device identifiers" can be strictly categorized as "personal information". Many policies do not consider "device identifiers" as "personal information".

PoliGraph also enables the analysis of purposes of data collection associated with data flows. Please refer to Appendix C in [19] for additional results.

6 Related Work

Formalizing Policies. A body of related work focuses on standardizing or formalizing policies. W3C P3P standard [37] proposed an XML schema to describe policies. The Contextual Integrity (CI) [35] framework expresses policies as information flows with parameters including the senders, recipients and subjects of information, data types, and transmission principles that describe the contexts of data collection. None of them replaces text-format policies, but they give insights into defining policies and serve as analysis frameworks. PoliGraph builds on the CI framework by extracting entities, data types, and part of the transmission principle (*i.e.,* purposes) from the policy text.

Policy Analysis. Another body of work analyzes policy text. OPP-115 [7] is a policy dataset with manual annotations for fine-grained data practices labeled by experts. Shvartzshnaider *et al.* [38], with the help of crowdsourced workers, analyze CI information flows extracted from policies to identify writing issues, such as incomplete context and vagueness. This manual approach is difficult to scale up for hundreds or thousands of policies due to the significant human efforts.

Automated Policy Analysis. The progress in NLP has made it possible to automate the analysis of unstructured text, such as policy text. Privee [39] uses binary text classifiers to answer whether a policy specifies certain privacy practices, such as data collection, encryption and ad tracking. Polisis [11], trained on the OPP-115 dataset, uses 10 multi-label text classifiers to identify data practices, such as the category of data types being discussed and purposes. Classifier-based methods use pre-defined labels which cannot capture the finer-grained semantics in the text. PolicyLint [9] first uses NLP linguistic analysis to extract data types and entities in collection statements. PurPliance [5], built on top of PolicyLint, further extracts purposes. Conceptually, both works focus on analyzing one sentence at a time, and extracting a tuple ⟨*entity, collect, data type*⟩, as well as *purpose* in PurPliance, albeit in a separate, nested tuple ⟨*data type, for / not_for,* ⟨*entity, purpose*⟩⟩. Unlike PoliGraph, these works view extracted tuples individually and do not infer data practices disclosed across multiple sentences.

Knowledge Graphs. Graphs are routinely used to integrate knowledge bases as relationships between terms [15]. Google has used a knowledge graph built from crawled data to show suggestions in search results [40]. OpenIE [41] and T2KG [16] use NLP to build knowledge graphs from a large corpus of unstructured text. In PoliGraph, we use knowledge graphs, for the first time, to represent policies.

7 Conclusion

Summary. We present PoliGraph, a framework that represents a privacy policy as a knowledge graph that (1) connects statements about data collection, use and sharing across different parts of the policy text, and (2) clearly defines and distinguishes between local and global ontologies. We design and implement PoliGraph-er—a tool that leverages NLP linguistic analysis to generate PoliGraphs from policy text. Because PoliGraph allows inferences of collection relations across paragraphs and sections, it significantly improves *recall* over prior work, while maintaining a high *precision* (see Section 4.2). Our manual validation shows that PoliGraph-er improves the recall of collection statements from 30% to 70% in comparison to PolicyLint [9]. Meanwhile, the precision is improved from 92% to 97%. PoliGraph, and the global ontologies used with it, also enable new policy analyses that were not previously possible: summarizing patterns in a corpus of policies (see Section 5.1) and assessing the correctness of definitions of terms (see Section 5.2). Our modular design also allows for extensions to the framework so as to analyze contradicting statements in more fine-grained contexts than prior work (see Section 5.3).

The source code of PoliGraph-er and the dataset used in this paper are made publicly available [42, 43].

Limitations. The proposed approach has its limitations. First, wrong edges in PoliGraph may propagate to wrong inferences (see Section 4.1.2). Second, despite our improvement in covering more sentence patterns than prior work (see Section 4.2), PoliGraph-er still faces the limitations of NLP. For example, it cannot parse tables and some lists, and misses a non-negligible number of collection statements. It is difficult to cover all possible syntactic patterns.

In the contradiction analysis (see Section 5.3.3), although we extend PoliGraph to encode additional context in policies (such as fine-grained action types and children as data subjects), we show that there are still false alarms due to the inevitability of handling *all* possible language nuances (*e.g.,* other types of data subjects, conditions, sources). Additional discussion can be found in Appendix D in [19].

Finally, as discussed in Section 4.1, given the high precision and improved, but imperfect, recall, we recommend to interpret what PoliGraph-er captures as a lower bound of the actual collection statements. The aforementioned limitations may unevenly affect different writing styles of policies (*e.g.,* text *vs.* tables), different sentence structures (*i.e.,* missing syntactic patterns), and even different applications. Therefore, the results of NLP analysis, especially results on each individual policy, should be validated by human experts.

Future Directions. Recent advancements in generative large language models (LLMs), represented by ChatGPT [44, 45], have greatly enhanced the capacity of NLP to understand complex text, easing the necessity for combining many NLP components and heuristics. However, the natural language interface of generative models can be a challenge for automated analysis. We expect that knowledge graphs will remain powerful tools to integrate model answers for automated analysis at scale. We will explore the potential of LLMs in policy analysis, to extend the PoliGraph framework so as to better incorporate additional elements in policy text, such as conditions of data collection and consent requirements.

Acknowledgments

This work is supported by NSF award 1956393 and a gift from the Noyce Initiative. We would like to thank our shepherd and reviewers from USENIX Security 2023 for their feedback, which helped to significantly improve the paper.

References

[1] Publications Office of the European Union, "General Data Protection Regulation (GDPR)." https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:02016R0679-20160504, 2016.

[2] State of California Department of Justice, "California Consumer Privacy Act of 2018 (amended by the California Privacy Rights Act of 2020)." https://leginfo.legislature.ca.gov/faces/codes_displayText.xhtml?lawCode=CIV&division=3.&title=1.81.5.&part=4., 2020.

[3] S. Jordan, S. Narasimhan, and J. Hong, "Deficiencies in the disclosures of privacy policy and in user choice," *Loyola Consumer Law Review*, vol. 34, p. 408–483, 2022.

[4] KAYAK, "KAYAK - Privacy Policy." https://www.kayak.com/privacy, July 2022.

[5] D. Bui, Y. Yao, K. G. Shin, J.-M. Choi, and J. Shin, "Consistency analysis of data-usage purposes in mobile apps," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, Association for Computing Machinery, 2021.

[6] R. Trimananda, H. Le, H. Cui, J. T. Ho, A. Shuba, and A. Markopoulou, "OVRseen: Auditing network traffic and privacy policies in oculus VR," in *31st USENIX Security Symposium (USENIX Security 22)*, USENIX Association, Aug. 2022.

[7] S. Wilson, F. Schaub, A. A. Dara, F. Liu, S. Cherivirala, P. G. Leon, M. S. Andersen, S. Zimmeck, K. M.

Sathyendra, N. C. Russell, *et al.*, "The creation and analysis of a website privacy policy corpus," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Aug. 2016.

[8] S. Wilson, F. Schaub, R. Ramanath, N. Sadeh, F. Liu, N. A. Smith, and F. Liu, "Crowdsourcing annotations for websites' privacy policies: Can it really work?," in *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, Apr. 2016.

[9] B. Andow, S. Y. Mahmud, W. Wang, J. Whitaker, W. Enck, B. Reaves, K. Singh, and T. Xie, "PolicyLint: Investigating internal privacy policy contradictions on google play," in *28th USENIX Security Symposium (USENIX Security 19)*, USENIX Association, Aug. 2019.

[10] B. Andow, S. Y. Mahmud, J. Whitaker, W. Enck, B. Reaves, K. Singh, and S. Egelman, "Actions speak louder than words: Entity-Sensitive privacy policy and data flow analysis with PoliCheck," in *29th USENIX Security Symposium (USENIX Security 20)*, USENIX Association, Aug. 2020.

[11] 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 (USENIX Security 18)*, USENIX Association, Aug. 2018.

[12] C. Lentzsch, S. J. Shah, B. Andow, M. Degeling, A. Das, and W. Enck, "Hey alexa, is this skill safe?: Taking a closer look at the alexa skill ecosystem," in *Network and Distributed Systems Security (NDSS) Symposium 2021*, Feb. 2021.

[13] U. Iqbal, P. N. Bahrami, R. Trimananda, H. Cui, A. Gamero-Garrido, D. Dubois, D. Choffnes, A. Markopoulou, F. Roesner, and Z. Shafiq, "Your echos are heard: Tracking, profiling, and ad targeting in the amazon smart speaker ecosystem," *arXiv preprint arXiv:2204.10920*, 2022.

[14] S. Manandhar, K. Kafle, B. Andow, K. Singh, and A. Nadkarni, "Smart home privacy policies demystified: A study of availability, content, and coverage," in *31st USENIX Security Symposium (USENIX Security 22)*, USENIX Association, Aug. 2022.

[15] G. A. Miller, "Wordnet: A lexical database for english," *Communications of the ACM*, vol. 38, p. 39–41, Nov. 1995.

[16] N. Kertkeidkachorn and R. Ichise, "T2kg: A demonstration of knowledge graph population from text and its challenges," in *Workshop and Poster Proceedings of the 8th Joint International Semantic Technology Conference*, 2018.

[17] Duck Duck Go, Inc., "duckduckgo/tracker-radar: Data set of top third party web domains with rich metadata about them." https://github.com/duckduckgo/tracker-radar, 2022.

[18] Peter Tripp (@notpeter), "notpeter/crunchbase-data: 2015 CrunchBase Data Export as CSV." https://github.com/notpeter/crunchbase-data, 2016.

[19] H. Cui, R. Trimananda, A. Markopoulou, and S. Jordan, "POLIGRAPH: Automated privacy policy analysis using knowledge graphs," *arXiv preprint arXiv:2210.06746*, 2023.

[20] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, "spaCy: Industrial-strength Natural Language Processing in Python." https://spacy.io/, 2020.

[21] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[22] Explosion, "spaCy - en_core_web_trf-3.5.0 pipeline." https://github.com/explosion/spacy-models/releases/tag/en_core_web_trf-3.5.0, 2023.

[23] M.-C. de Marneffe, T. Dozat, N. Silveira, K. Haverinen, F. Ginter, J. Nivre, and C. D. Manning, "Universal Stanford dependencies: A cross-linguistic typology," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, European Language Resources Association (ELRA), May 2014.

[24] Explosion, "Rule-based matching - spaCy Usage Documentation." https://spacy.io/usage/rule-based-matching, 2023.

[25] M. Joshi, O. Levy, L. Zettlemoyer, and D. Weld, "BERT for coreference resolution: Baselines and analysis," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Association for Computational Linguistics, Nov. 2019.

[26] R. P. Hudson, "msg-systems/coreferee: Coreference resolution for English, French, German and Polish, optimised for limited training data and easily extensible for further languages." https://github.com/msg-systems/coreferee, 2022.

[27] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Nov. 2019.

[28] L. Tunstall, N. Reimers, U. E. S. Jo, L. Bates, D. Korat, M. Wasserblat, and O. Pereg, "Efficient few-shot learning without prompts," *arXiv preprint arXiv:2209.11055*, 2022.

[29] Pixel Tale Games, ""Puzzle 100 Doors" Privacy Policy." https://proteygames.github.io/, July 2022.

[30] Microsoft, "Fast and reliable end-to-end testing for modern web apps | Playwright." https://playwright.dev/, 2022.

[31] Paleblue Corp., "Paleblue Corp. Privacy Policy." http://nalst.cafe24.com/policy.html, 2015.

[32] O. Ellenbogen, "Manager Readme Privacy Policy." https://managerreadme.com/privacy, 2022.

[33] Armor Games Inc., "Armor Games Inc. Privacy Policy." https://armorgamesstudios.com/privacy/, Jan. 2020.

[34] Horizone Media, "Horizone Media: Privacy Policy." http://horizonemedia2018.blogspot.in/2018/01/privacy-policy.html, Jan. 2018.

[35] H. Nissenbaum, *Privacy in Context - Technology, Policy, and the Integrity of Social Life*. Stanford University Press, 2009.

[36] Internet Archive: Wayback Machine, "Internet Archive." https://archive.org/web/, 2023.

[37] World Wide Web Consortium and others, "The Platform for Privacy Preferences 1.1 (P3P1.1) Specification." https://www.w3.org/TR/2018/NOTE-P3P11-20180830/, 2018.

[38] Y. Shvartzshnaider, N. Apthorpe, N. Feamster, and H. Nissenbaum, "Going against the (appropriate) flow: A contextual integrity approach to privacy policy analysis," in *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, Association for the Advancement of Artificial Intelligence, Oct. 2019.

[39] S. Zimmeck and S. M. Bellovin, "Privee: An architecture for automatically analyzing web privacy policies," in *23rd USENIX Security Symposium (USENIX Security 14)*, USENIX Association, Aug. 2014.

[40] Google, "Introducing the Knowledge Graph: things, not strings." https://blog.google/products/search/introducing-knowledge-graph-things-not/, 2012.

[41] G. Angeli, M. J. J. Premkumar, and C. D. Manning, "Leveraging linguistic structure for open domain information extraction," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 2015.

[42] UCI Networking Group, "POLIGRAPH GitHub Repository (including the Source Code of POLIGRAPH-ER)." https://github.com/UCI-Networking-Group/PoliGraph, June 2023.

[43] UCI Networking Group, "Policy-Technology | UCI Networking Group." https://athinagroup.eng.uci.edu/projects/auditing-and-policy-analysis/, June 2023.

[44] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," *arXiv preprint arXiv:2203.02155*, 2022.

[45] OpenAI, "GPT-4 Technical Report," *arXiv preprint arXiv:2303.08774*, 2023.