The Privacy Quagmire: Where Computer Scientists and Lawyers May Disagree

Yunwei Zhao[1], Varun Chandrasekaran[2], Thomas Wies[1], Lakshmi Subramanian[1]
[1]New York University [2]University of Illinois Urbana-Champaign

Abstract

Privacy policies dictate how systems handle user data, yet engineers struggle to verify compliance because policies use intentionally vague legal language. Current automated analyzers extract data practices using NLP but fail when policies say things like "share data for legitimate purposes" - terms that have no computational definition. This mismatch between legal flexibility and formal verification creates a fundamental barrier to automated compliance checking. We identify four systematic challenges: vague terms, evolving terminology, exception patterns that appear contradictory, and external legal dependencies. We propose an approach that preserves this ambiguity, where we use LLMs to extract structured parameters and convert them to first-order logic while keeping vague conditions as explicit placeholders for human interpretation. Our system can extract hundreds of data practices and reveals hidden complexities in TikTok and Meta policies, though the resulting formulas remain too complex for SMT solvers. This demonstrates the promise and fundamental limits of formalizing the legal text.

CCS Concepts

• **Security and privacy → Logic and verification**; **Privacy protections**.

Keywords

Knowledge Graphs, Formal Verification, Data Privacy

ACM Reference Format:
Yunwei Zhao[1], Varun Chandrasekaran[2], Thomas Wies[1], Lakshmi Subramanian[1]. 2025. The Privacy Quagmire: Where Computer Scientists and Lawyers May Disagree. In *The 24th ACM Workshop on Hot Topics in Networks (HotNets '25), November 17–18, 2025, College Park, MD, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3772356.3772422

HotNets '25, College Park, MD, USA

ACM ISBN 979-8-4007-2280-6/2025/11
https://doi.org/10.1145/3772356.3772422

1 Introduction

Lawyer: *"Our privacy policy is clear: We never share personal data with anyone, except to comply with the law or with the user's express consent."*
Computer Scientist: *"If I translate that into code, I get one rule saying 'never share any personal data,' and another rule saying 'it's okay to share personal data if required by law or if the user consents.' My static analyzer flags a contradiction because it doesn't know what 'required by law' covers or how to handle user consent."*
Lawyer: *"There's no contradiction for a human reader - the policy means we generally do not share data, but we make an exception when legally compelled or when the user agrees."*
Computer Scientist: *"Exactly. To a program, those exceptions are undefined terms. Without defining which law or how consent is given, the policy's meaning is incomplete from a formal perspective."*

This exchange shows the fundamental gap between legal language and code verification. Privacy policies govern how organizations handle personal data: who can access it, share it, and under what conditions. From a computer science perspective, privacy protection often relies on cryptographic guarantees like differential privacy [6] for statistical disclosure control or secure multi-party computation [16] for confidential data processing. But real privacy policies address broader concerns than data confidentiality - they encode legal requirements about appropriate data use, business practices, and user expectations that resist simple formalization. Contextual Integrity [12] offers an alternative framework, arguing that privacy violations occur when information flows inappropriately between contexts according to social norms, which remain difficult to formalize computationally.

Modern systems enforce privacy policies through multiple technical mechanisms: encryption protects data confidentiality, access controls restrict who can view data, and runtime monitors like TaintDroid [7] track how data flows through applications. These mechanisms effectively enforce well-defined rules - "only users with role X can access database Y." But they struggle with ambiguous policy statements like *"share data only with trusted partners for business purposes."* Developers must somehow translate these vague terms into

concrete access control rules. New regulations like GDPR and CCPA compound this problem by adding jurisdiction-specific requirements that change how data can be processed. The Usable Privacy Policy Project [14] documented that such vague language appears in over 75% of privacy policies, making automated enforcement challenging.

To bridge the gap between legal text and technical enforcement, researchers have developed automated policy analysis tools. These systems aim to extract machine-readable rules from privacy policies that could inform access control decisions or detect policy violations. Recent work has used semantic parsing [11] to convert natural language to logic, knowledge graphs [9] to represent relationships between data practices, and neural models to classify policy statements. Earlier work on semantic role labeling [13] showed that extracting semantic roles - who does what to whose data under which conditions - improves accuracy over simple classification. PoliGraph [5] represents policies as knowledge graphs where nodes are entities and edges are data practices. PolicyLR [10] uses predefined taxonomies to classify statements, while NL2FOL [11] directly converts statements to logic without fixed categories.

Grammar-based parsers fail when "share data with service providers" and "provide information to third parties" are semantically equivalent with different syntax. Neural parsers would force vague terms such as "legitimate purposes" into predefined categories. We thus use LLMs to extract semantic roles and organize data types into hierarchies while preserving vague terms as uninterpreted predicates for human interpretation.

1.1 Challenges

We outline four fundamental challenges that act as roadblocks to making fully automated privacy compliance checking feasible.

The first challenge is ***vague language***. Privacy policies use terms that have no computational definition. When a policy states *"We share data with service providers for business operations,"* what exactly counts as *"business operations"*? Payroll processing? Marketing analytics? Customer support? Formal verification requires precise predicates, but these terms are intentionally flexible to cover future business needs. Recent approaches [5, 9, 13] can identify that a statement involves data sharing and extract the involved parties, but they cannot define what "business operations" means in computational terms.

The second challenge is ***evolving terminology***. Privacy policies constantly introduce new concepts - biometric templates, blockchain addresses, AI training datasets - that don't fit into predefined categories. Existing tools rely on fixed categorization schemes (predefined sets of data types and purposes) to classify policy statements. For example, PolisiS [9] uses the OPP-115 taxonomy that includes categories like "financial info" and "health info." But when policies mention novel concepts like "neural network embeddings," these fixed taxonomies fail. Attempting to dynamically expand taxonomies [10] risks creating overlapping categories - is "facial recognition data" a type of "biometric data" or "image data"? Such overlaps break the logical consistency needed for formal reasoning. Even MAPS [18], which analyzed over one million Android apps, struggles when apps collect data types that weren't common when the system was trained.

The third challenge is ***recognizing exceptions***. Policies state general rules then carve out specific exceptions. *"We don't share location data"* followed later by *"We share location data with mapping services"* appears contradictory to automated tools that treat each statement independently. But humans understand the second statement creates a specific exception to the general rule. Current analyzers can detect that both statements mention location data and sharing [11], but they struggle to recognize the hierarchical relationship where specific rules override general ones. PolicyLint [1] found that 14.2% of apps contain such apparent contradictions; manual review revealed most were actually coherent exception patterns.

The fourth challenge is ***external dependencies***. Policies reference external context that isn't defined within the policy text. *"We share data when required by law"* depends on which laws apply in each jurisdiction. *"We notify users through their account settings"* depends on the application's actual implementation of those settings. Formalizing these statements requires information beyond the policy text itself. Entity-sensitive analysis like PoliCheck [2] can track that "law enforcement" is an entity that receives data, but it cannot determine which specific laws trigger sharing in which contexts.

1.2 Our Approach in a Nutshell

In this paper, we propose an approach that embraces these limitations rather than abstracting them away. We extract structured information from privacy policies while preserving ambiguity for human review. Specifically, we use an LLM to identify six key elements in each policy statement: the data sender, receiver, data subject (whose data it is), data type, action performed, and any conditions. We chose these elements based on Contextual Integrity theory [13], which models privacy as appropriate information flow. However, we do not attempt to extract full "transmission principles" (the complex social norms governing when data sharing is appropriate), as these require human judgment about social context. Unlike PolicyLR [10] which relies on fixed taxonomies, we build hierarchies dynamically. Unlike NL2FOL [11] which converts

individual statements, we maintain global context through hierarchical graphs.

We encode the extracted elements as first-order logic (FOL) formulas because FOL provides a standard formalism for expressing relationships and constraints that can be analyzed by automated theorem provers. Unlike knowledge graphs that only capture relationships, FOL can express complex conditions like "for all users X, if X has opted out, then do not share X's data." When we encounter vague terms like "legitimate purposes," we preserve them as uninterpreted predicates rather than attempting to define them, making the ambiguity explicit.

2 System

Our system transforms natural language privacy statements into formal first-order logic (FOL) representations by extracting their key components—the entities, data, and relationships that define each data practice. Specifically, we identify the information `sender`, `receiver`, `subject`, and `attribute` involved in each data practice, as well as the `action` (the verb describing the data practice) and any contextual `condition` under which the action occurs. This approach adapts elements from Contextual Integrity theory [12]. However, we do not attempt to extract full transmission principles. Prior work [13] has shown these require significant human interpretation. This structured representation enables detection of policy conflicts and incomplete disclaimers. The pipeline consists of three phases that directly address the challenges identified in Section 1.

Phase 1: LLM-based Representation Extraction. To handle vague language (Challenge 1), we use large language models with few-shot prompting to extract semantic roles from policy text. A semantic role identifies the function each entity plays in a data practice—for example, in "TikTok shares data with advertisers," TikTok fills the sender role while advertisers fill the receiver role. Unlike rule-based parsers that fail on ambiguous phrasing, LLMs can interpret varied expressions of the same concept. This builds on recent successes in legal text analysis [4]. We resolve coreferences by using the LLM to identify that pronouns like "we" and "our" refer to the company name extracted from the policy's opening. We segment policies into individual statements about data practices. Each segment receives a unique identifier based on its content hash, which enables diff-based tracking when policies update. Only modified segments require re-extraction. For each segment, we extract structured parameters that capture who shares what data with whom under which conditions. Crucially, when the LLM encounters vague terms like *"legitimate business purposes,"* these are preserved as-is in the condition field. This makes the ambiguity explicit rather than forcing premature interpretation. The hierarchical relationships built in Phase 2 can later help clarify these vague terms by identifying specific instances that reveal what the broad category encompasses.

Phase 2: Hierarchical Graph Construction. To address evolving terminology (Challenge 2), we build dynamic hierarchies using Chain-of-Layer (CoL) prompting [17] rather than relying on fixed taxonomies. Unlike previous taxonomy-based approaches [15] that struggle with novel concepts, CoL constructs taxonomies iteratively. This approach requires no domain-specific knowledge and can adapt to healthcare, media, financial, or educational terminology through the same iterative process. Starting with a root concept (e.g., "data"), it identifies immediate subcategories in the first layer (e.g., "personal data," "technical data"), then builds subsequent layers by finding subcategories of each node (e.g., "email" under "personal data"). This ensures all terms are incorporated while maintaining semantic consistency. This produces two structures: a data hierarchy organizing data types by subsumption relationships, and an entity-data graph capturing who performs which actions on what data. Both structures persist across policy versions. When text changes, we identify affected nodes through segment tracking and update only those branches. Conditions under which actions occur are modeled as boolean predicates attached to edges in the entity-data graph. When conditions reference other entities, we extract these as additional relationships while keeping the conditions themselves as logical constraints. This separation allows the system to handle novel data types through dynamic hierarchy construction. It also properly recognizes exceptions (Challenge 3) by maintaining conditions as explicit constraints on when rules apply.

Phase 3: Semantic Query Verification. To handle both recognizing exceptions (Challenge 3) and external dependencies (Challenge 4), we employ a hybrid approach combining neural embeddings with formal logic. When a user poses a query, we first translate it to policy vocabulary using embedding-based similarity search. This bridges the vocabulary gap between how users phrase questions and how policies state rules. The approach is similar to that in Polisis [9] but maintains logical structure. We then extract relevant subgraphs that include both direct matches and hierarchically related terms. For new queries, we construct the subgraph by reusing existing hierarchy and do local graph traversal rather than full reconstruction. For instance, if a policy allows sharing *"contact information"* and we know *"email address"* is a subtype, the hierarchy enables proper inference. Conditions and external references are converted to boolean predicates. Undefined terms become named predicates that make incompleteness explicit. The final FOL formula is checked by an SMT solver to provide formal verification where possible. When the formula contains undefined predicates like

`required_by_law` or `legitimate_business_purpose`, these appear as uninterpreted symbols in the verification. This signals that the result depends on how these vague terms are resolved. It effectively identifies where human judgment or external input is necessary to complete the analysis.

This three-phase design directly addresses our core challenges. Vague language is handled through LLM interpretation with explicit preservation of ambiguity. Evolving terminology is managed through dynamic hierarchy construction. Exception recognition leverages hierarchical relationships in our graphs. External dependencies become explicit predicates in our formal representation. Rather than attempting full automation, our system provides a structured framework where formal methods identify clear-cut issues while human expertise resolves genuine ambiguities.

3 Implementation

We implemented our system in Python using GPT-4o-mini for extracting semantic roles (sender, receiver, action, etc.) from policy text, text-embedding-3-large for computing similarity between terms, and CVC5 for solving logical formulas. This section describes the key algorithms and implementation decisions.

Algorithm 1 shows our complete pipeline. The implementation makes several key design decisions to handle real-world privacy policies effectively.

Company Name Extraction and Coreference Resolution. We extract the company name from the policy by prompting the LLM to identify the organization name in the first 1000 characters. This enables consistent coreference resolution throughout the document. We replace pronouns like "we," "us," and "our" with the actual company name. Each segment maintains context from previous segments to resolve ambiguous references. For example, when a policy says "we collect your data," we replace "we" with "TikTok" based on the extracted company name.

Semantic Parameter Extraction. The extraction prompt includes specific normalization rules: actions are converted to base form ("collects" becomes "collect"), data types are singularized ("email addresses" becomes "email address"), and user references are standardized to "user". The prompt includes few-shot examples demonstrating how to handle compound statements by generating multiple sets of parameters (one for each data practice mentioned). Conditions are extracted to capture only the circumstances under which actions occur, not the main action itself. We preserve logical operators like AND/OR. This normalization approach follows best practices from semantic role labeling for privacy texts [13].

Chain-of-Layer Hierarchy Construction. Following Chain-of-Layer [17], we implement iterative taxonomy building. The algorithm first identifies a root concept using an LLM prompt, then builds the taxonomy layer by layer. At each iteration, it identifies next-level entities from the remaining set and establishes their relationships to existing nodes. We optionally filter out unlikely relationships by computing SciBERT [3] similarity scores between terms and removing pairs with similarity below a threshold. The algorithm ensures every entity appears exactly once in the final taxonomy while maintaining hierarchical consistency.

Embedding-Based Semantic Search. We pre-compute vector representations (embeddings) for all graph elements using OpenAI's text-embedding-3-large model. For query processing, we implement a multi-step translation process. First, we compute cosine similarity between query terms and all policy terms. For top-k (k=10) high-similarity pairs, we verify semantic equivalence using an LLM prompt. The prompt asks whether the terms mean the same in a privacy context. The search extends beyond individual terms to edge representations. We embed the concatenation of source, action, and target for more accurate matching.

FOL Formula Generation. The conversion to first-order logic creates structured formulas where query constraints become expressions with existential quantifiers ("there exists some X such that...") and policy statements become logical disjunctions (OR operations) of permitted actions. Conditions are converted to boolean predicates (true/false functions). Vague conditions become named predicates that expose their undefined nature. The formula checks whether the query logically follows from the policy statements. This approach adapts techniques from NL2FOL [8] to the privacy domain.

SMT-LIB Generation and Solving. We implement a custom compiler that converts FOL formulas to SMT-LIB v2 format (the standard input language for SMT solvers). The compiler extracts all predicates and constants from the formula, generates proper declarations, handles variable scoping in quantified expressions, and asserts the negation of the implication for checking logical validity. We use CVC5 as our SMT (Satisfiability Modulo Theories) solver - a tool that determines whether logical formulas are satisfiable. We interpret "unsat" results as valid (the query necessarily follows from the policy) and "sat" results as invalid (the query does not necessarily follow).

The implementation includes caching mechanisms for segments, extracted parameters, graphs, and embeddings to enable incremental processing and debugging. All intermediate representations are stored in JSON or pickle format. This allows inspection of each pipeline stage.

Algorithm 1 Privacy Policy FOL Extraction and Query Verification

Input: Policy text $\mathcal{P}$, OPP-115 taxonomy $\mathcal{T}$, User query Q
Output: *result* $\in \{VALID, INVALID, UNKNOWN\}$
Parameters: θ = sender, ρ = receiver, κ = subject, π = data type, α = action, c = condition, p = permission

// Phase 1: Extract Representations
$C \leftarrow \text{ExtractCompanyName}(\mathcal{P})$
$\mathcal{P}' \leftarrow \text{ResolveCoreferences}(\mathcal{P}, C)$
$S \leftarrow \text{Segment}(\mathcal{P}')$ // $S = \{s_1, s_2, ..., s_n\}$
$\mathcal{E} \leftarrow \emptyset$
for $s \in S$ **do**
  $(\theta, \rho, \kappa, \pi, \alpha, c, p) \leftarrow \text{ExtractParams}(s, C)$
  $\mathcal{D} \leftarrow \text{Match}(s, \mathcal{T})$ // Identify data types
  $\mathcal{E} \leftarrow \mathcal{E} \cup \{(\theta, \rho, \kappa, \pi, \alpha, c, p, \mathcal{D})\}$
end for
// Phase 2: Build FOL Graphs
entities $\leftarrow \text{ExtractEntities}(\mathcal{E})$
data types $\leftarrow \text{ExtractDataTypes}(\mathcal{E})$
$H_E \leftarrow \text{ChainOfLayer}(\text{entities}, \text{“entity”})$
$H_D \leftarrow \text{ChainOfLayer}(\text{data types}, \text{“data”})$
$G_{\text{ED}}, G_{\text{DD}} \leftarrow \text{BuildGraphs}(\mathcal{E}, H_E, H_D)$
embeddings $\leftarrow \text{ComputeEmbeddings}(G_{\text{ED}}, G_{\text{DD}})$
// Phase 3: Query Verification
$Q' \leftarrow \text{ResolveCoreferences}(Q, C)$
$q \leftarrow \text{ExtractParams}(Q', C)$ // Extract query params
$q_{trans} \leftarrow \text{TranslateTerms}(q, \text{embeddings})$
matches $\leftarrow \text{SemanticSearch}(q_{trans}, G_{\text{ED}}, \text{embeddings})$
subgraph $\leftarrow \text{BuildSubgraph}(q_{trans}, \text{matches}, G_{\text{DD}})$
$\Phi \leftarrow \text{ToFOL}(q_{trans}, \text{subgraph})$
result $\leftarrow \text{SMT}(\neg\Phi)$

return *result*

4 Results

We evaluated our system on privacy policies from two major technology companies: TikTok and Meta. These policies represent different scales of complexity - TikTok's policy contains approximately 15,000 words while Meta's spans over 40,000 words. This section presents qualitative analysis of the extraction quality and verification capabilities.

4.1 Extraction Results

Table 1 summarizes the extraction results for both policies. The system successfully processed both policies, demonstrating scalability from shorter to more comprehensive privacy documents.

Table 1: Extraction Statistics for Privacy Policies

Metric	TikTok	Meta
Total nodes	419	1,323
Total edges	974	3,801
Entities	217	700
Data types	122	382

4.2 Quality Analysis: TikTok Policy

Our system extracted 974 distinct data practice edges from TikTok's policy. Each edge represents a directed relationship between entities (e.g., `[user] -provide-> [email]`). Table 2 presents representative examples demonstrating how single policy statements generate multiple such relationships.

The decomposition reveals several key insights. First, the system identifies multiple actors and actions within single statements. The contact-finding example shows both user actions (choosing to find users) and TikTok's resulting collection activities, capturing the causal relationship. Second, enumerated lists are properly expanded - the profile information statement generates ten distinct edges, ensuring each data type is individually trackable. This granular extraction enables precise queries about specific data types rather than broad categories. The system uses embedding-based similarity to match related terms; for instance, "email address" in a query matches to "email" in the policy with 0.999 similarity.

The system also preserves implicit relationships. When TikTok states it will "access and collect" contact information, the extraction captures both the intermediate access step and the final collection, revealing the data flow process. This level of detail is crucial for compliance analysis, as regulations often distinguish between data access and data retention.

4.3 Quality Analysis: Meta Policy

Meta's more comprehensive policy yielded 3,801 edges with notably more complex data flows. Table 3 shows how the system handles Meta's intricate policy language.

Meta's extractions demonstrate the system's ability to handle sophisticated data practices. The camera/voice features example shows six distinct edges capturing different aspects of multimedia data collection. Notably, the system distinguishes between content provided through different mechanisms (camera feature vs. voice features) and identifies both user provision and Meta's collection as separate actions.

The interaction tracking example reveals Meta's comprehensive monitoring, generating six edges that capture viewing, interacting, and engagement as distinct activities. This granularity is essential for understanding the full scope of

Policy Statement	Extracted Edges
When you create an account, upload content, contact TikTok directly, or otherwise use the Platform, you may provide some or all of the following information	[user]-create->[account], [user]-upload->[content], [user]-contact->[TikTok directly], [user]-use->[Platform], [user]-provide->[information]
Account and profile information, such as name, age, username, password, language, email, phone number, social media account information, and profile image	[user]-create->[account and profile], [user]-provide->[name], [user]-provide->[age], [user]-provide->[username], [user]-provide->[password], [user]-provide->[language], [user]-provide->[email], [user]-provide->[phone number], [user]-provide->[social media account information], [user]-provide->[profile image]
If you choose to find other users through your phone contacts, TikTok will access and collect information such as names, phone numbers, and email addresses	[user]-choose to find->[other users through phone contacts], [user]-access->[phone contacts], [TikTok]-collect->[contact list], [TikTok]-collect->[name of contacts], [TikTok]-collect->[phone number of contacts], [TikTok]-collect->[email address of contacts]

Table 2: TikTok policy statements decomposed into multiple semantic edges

Policy Statement	Extracted Edges
Content you provide through Meta's camera feature or your camera roll settings, or through Meta's voice-enabled features	[user]-provide->[camera feature content], [user]-provide->[voice-enabled features content], [user]-allow access->[photos and videos], [user]-give->[photos and videos], [Meta]-collect->[information from camera], [Meta]-collect->[information about how user uses Camera feature]
Types of content, including ads, you view or interact with, and how you interact with it	[user]-view->[content], [user]-interact with->[content], [user]-view->[ads], [user]-interact with->[ads], [user]-engage with->[ads and commercial content], [user]-provide->[interaction data]
Purchases or other transactions you make, such as through Meta checkout experiences, including credit card information	[user]-make->[purchases], [user]-make->[transactions], [user]-provide->[truncated credit card information], [user]-make->[payments using Meta Pay], [user]-make->[purchases in Marketplace], [user]-make->[purchases within online game], [Meta]-process->[financial information], [Meta]-access->[financial transaction data], [Meta]-preserve->[financial transaction data]

Table 3: Meta policy statements revealing complex multi-actor data flows

behavioral tracking. The system correctly identifies viewing and interacting as separate trackable actions on both general content and advertisements specifically. Through embedding similarity, queries about "location data" successfully match related terms like "location information" and "GPS location" in the policy.

The financial transaction example is particularly revealing, with nine edges capturing the complete payment ecosystem. The extraction distinguishes between different payment contexts (Meta Pay, Marketplace, in-game purchases) and different stages of financial data handling (collection, processing, preservation). This comprehensive extraction enables queries about specific payment scenarios that might have different privacy implications.

4.4 Addressing Key Challenges

The multi-edge extraction approach directly addresses our identified challenges:

Vague Language (Challenge 1). By decomposing statements into individual relationships, vague terms become isolated and explicit. When Meta mentions collecting "information about how you use Camera feature," the edge preserves this vagueness while making it queryable.

Novel Terminology (Challenge 2). New concepts like "voice-enabled features" and "Meta checkout experiences"

are captured as distinct nodes, automatically incorporated into the graph without predefined categories.

Exception Handling (Challenge 3). Conditional actions are captured through additional metadata on edges. The TikTok contact example shows how user choice ("if you choose to find other users") triggers specific collection activities.

External Dependencies (Challenge 4). Complex multi-actor scenarios are decomposed into individual edges, making dependencies explicit. The Meta payment example shows how user purchases trigger Meta's processing and preservation actions.

4.5 Scalability Evaluation

The successful generation of valid SMT-LIB formulas (the standard format for SMT solvers) for both policies validates our approach of combining neural language models with formal methods. The semantic search phase successfully addresses verification scalability by reducing the search space before formal encoding. Without this phase, a query would require the SMT solver to reason about thousands of disjunctive clauses, which leads to exponential complexity. The semantic search reduces each query to a small subset of relevant edges: 6.4 edges on average for TikTok and 18.5 edges on average for Meta, representing 99.38% and 99.50% reduction in the verification problem respectively. The SMT solver then operates on this filtered subset, which enables tractable formal reasoning. Empirical evaluation on both policies (TikTok: 974 edges, Meta: 3,801 edges) across 23 queries of varying complexity achieved zero timeouts with average query times of 3.39s and 3.91s respectively. This demonstrates sub-linear scaling behavior: Meta's policy is 3.9× larger than TikTok's policy, yet query times increased by only 1.15×. The structured graph representation enables compliance analysis. Legal teams can identify gaps and contradictions between policies, while engineers can extract concrete implementation requirements for privacy safety. The graph representation enables systematic comparison across policies and tracking of data flows through multiple actors.

5 Discussion

Combining LLMs with formal methods provides a systematic approach for analyzing privacy policies. While existing approaches focus on shallow text classification or assume simplified policy representations, real policies contain vague terms like "legitimate business purpose" that resist straightforward formalization.

We demonstrated that privacy policies can be systematically extracted and verified at scale. Processing TikTok and Meta policies yielded 974 and 3,801 data practice edges respectively. The semantic search optimization achieved zero timeouts across 23 queries with sub-linear scaling behavior. The hierarchical organization enables queries to match related terms across different phrasings.

The system serves four user groups. Policy authors track changes when updating for regulations and identify contradictions between versions. Compliance teams test privacy policies against scenarios to ensure consistency. Users query whether data handling complies with stated policies. Engineers extract logic and conditions needed to implement privacy-safe systems. Each group benefits from the hierarchical relationships and compliance verification.

Our multi-edge extraction enables scalable analysis of complex privacy policies. This approach handles the inherent challenges of policy language: vague terminology, novel concepts, exception handling, and external dependencies. The resulting graph representation enables automated verification and supports manual compliance analysis. This supports policy authors, compliance teams, end users, and engineers working with diverse privacy compliance and implementation tasks.

6 Limitations and Future Work

Multi-policy analysis. The current system analyzes policies independently. Organizations often maintain multiple privacy policies across different products, platforms, or jurisdictions. Extending the system to detect contradictions between related policies could help maintain consistency across an organization's privacy ecosystem. This would require resolving semantic differences in terminology across policies while preserving policy-specific nuances.

Cross-domain reasoning. The system currently extracts each policy into its own graph structure. Verifying compliance between policies and regulations requires aligning their representations, as different documents may use distinct terminology and taxonomies for the same concepts. Future work could develop methods to merge or align graphs across domains to enable automated compliance checking against regulations.

Temporal policy tracking. The system design supports incremental updates through content hashing: each segment has a unique hash, and edges are tracked to their source segments. When a policy updates, the system identifies changed segments and re-extracts only affected edges. Evaluating this mechanism on real policy revisions could reveal how policy changes propagate through the graph structure.

Acknowledgements

We thank our shepherd Paul Schmitt and the reviewers for their valuable comments and suggestions to improve the paper. Research reported in this publication was supported by an Amazon Research Award, Fall 2023.

References

[1] Benjamin Andow, Samin Yaseer Mahmud, Wenyu Wang, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Tao Xie. 2019. PolicyLint: Investigating Internal Privacy Policy Contradictions on Google Play. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 585–602. https://www.usenix.org/conference/usenixsecurity19/presentation/andow

[2] Benjamin Andow, Samin Yaseer Mahmud, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Serge Egelman. 2020. 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, 985–1002. https://www.usenix.org/conference/usenixsecurity20/presentation/andow

[3] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A Pretrained Language Model for Scientific Text. 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)*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Hong Kong, China, 3615–3620. doi:10.18653/v1/D19-1371

[4] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. [n. d.]. LEGAL-BERT: The Muppets straight out of Law School. In *Findings of EMNLP 2020*. 2898–2904. doi:10.18653/v1/2020.findings-emnlp.261

[5] Hao Cui, Rahmadi Trimananda, Athina Markopoulou, and Scott Jordan. 2023. PoliGraph: Automated Privacy Policy Analysis using Knowledge Graphs. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, 1037–1054. https://www.usenix.org/conference/usenixsecurity23/presentation/cui

[6] Cynthia Dwork. 2008. Differential Privacy: A Survey of Results. In *Theory and Applications of Models of Computation*. https://api.semanticscholar.org/CorpusID:2887752

[7] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick Mcdaniel, and Anmol Sheth. 2010. TaintDroid. *ACM Transactions on Computer Systems (TOCS)* 32 (2010), 1 – 29. https://api.semanticscholar.org/CorpusID:433048

[8] Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, Ekaterina Zubova, Yujie Qiao, Matthew Burtell, David Peng, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Shafiq R. Joty, Alexander R. Fabbri, Wojciech Kryscinski, Xi Victoria Lin, Caiming Xiong, and Dragomir R. Radev. 2022. FOLIO: Natural Language Reasoning with First-Order Logic. *ArXiv* abs/2209.00840 (2022). https://api.semanticscholar.org/CorpusID:252070866

[9] Hamza Harkous, Kassem Fawaz, Rémi Lebret, Florian Schaub, Kang G. Shin, and Karl Aberer. 2018. Polisis: Automated Analysis and Presentation of Privacy Policies Using Deep Learning. *ArXiv* abs/1802.02561 (2018). https://api.semanticscholar.org/CorpusID:3656756

[10] Ashish Hooda, Rishabh Khandelwal, Prasad R. Chalasani, Kassem Fawaz, and Somesh Jha. 2024. PolicyLR: A Logic Representation For Privacy Policies. *ArXiv* abs/2408.14830 (2024). https://api.semanticscholar.org/CorpusID:271962872

[11] Abhinav Lalwani, Lovish Chopra, Christopher Hahn, Caroline Trippel, Zhijing Jin, and Mrinmaya Sachan. 2024. NL2FOL: Translating Natural Language to First-Order Logic for Logical Fallacy Detection. *ArXiv* abs/2405.02318 (2024). https://api.semanticscholar.org/CorpusID:269604995

[12] Helen Nissenbaum. 2004. Privacy as contextual integrity. *Washington Law Review* 79 (2004), 119–157. https://api.semanticscholar.org/CorpusID:150528892

[13] Yan Shvartzshanider, Ananth Balashankar, Thomas Wies, and Lakshminarayanan Subramanian. [n. d.]. Beyond The Text: Analysis of Privacy Statements through Syntactic and Semantic Role Labeling. In *Natural Legal Language Processing Workshop at ACL 2023*.

[14] Shomir Wilson, Florian Schaub, Aswarth Abhilash Dara, Frederick Liu, Sushain Cherivirala, Pedro Giovanni Leon, Mads Schaarup Andersen, Sebastian Zimmeck, Kanthashree Mysore Sathyendra, N. Cameron Russell, Thomas B. Norton, Eduard Hovy, Joel Reidenberg, and Norman Sadeh. 2016. The Creation and Analysis of a Website Privacy Policy Corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Katrin Erk and Noah A. Smith (Eds.). Association for Computational Linguistics, Berlin, Germany, 1330–1340. doi:10.18653/v1/P16-1126

[15] Shomir Wilson, Florian Schaub, Aswarth Abhilash Dara, Frederick Liu, Sushain Cherivirala, Pedro Giovanni Leon, Mads Schaarup Andersen, Sebastian Zimmeck, Kanthashree Mysore Sathyendra, N. Cameron Russell, Thomas B. Norton, Eduard H. Hovy, Joel R. Reidenberg, and Norman M. Sadeh. 2016. The Creation and Analysis of a Website Privacy Policy Corpus. In *Annual Meeting of the Association for Computational Linguistics*. https://api.semanticscholar.org/CorpusID:15817938

[16] Andrew Chi-Chih Yao. 1982. Protocols for secure computations. *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)* (1982), 160–164. https://api.semanticscholar.org/CorpusID:62613325

[17] Qingkai Zeng, Yuyang Bai, Zhaoxuan Tan, Shangbin Feng, Zhenwen Liang, Zhihan Zhang, and Meng Jiang. 2024. Chain-of-Layer: Iteratively Prompting Large Language Models for Taxonomy Induction from Limited Examples. *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management* (2024). https://api.semanticscholar.org/CorpusID:267627600

[18] Sebastian Zimmeck, Peter Story, Daniel Smullen, Abhilasha Ravichander, Ziqi Wang, Joel R. Reidenberg, N. Cameron Russell, and Norman M. Sadeh. 2019. MAPS: Scaling Privacy Compliance Analysis to a Million Apps. *Proceedings on Privacy Enhancing Technologies* 2019 (2019), 66 – 86. https://api.semanticscholar.org/CorpusID:198490131