

THE RESEARCH UNIVERSITY PRESS

International
Computational Law Journal

VOLUME 2 / ISSUE 1 / CURATED SOURCE EDITION

The Dawn of a New Era of Compliance: Automated Compliance Verification and Enforcement

Markus Sobkowski · Gevorg Karapetyan

MIT Computational Law Report · Originally published 2025-02-24

Independent reader edition under CC BY 4.0. This attribution cover was added; original source PDF pages follow unchanged. Original authorship and source identity are retained. MIT, Stanford University, Stanford Law School and CodeX have not sponsored or endorsed this republication.

Source, license and version evidence accompany the online article.

MIT Computational Law Report

The Dawn of a New Era of Compliance: Automated Compliance Verification and Enforcement

Markus Sobkowski Gevorg Karapetyan

Published on: Feb 20, 2025

URL: https://law.mit.edu/pub/thedawnofaneweraofcompliance

License: Creative Commons Attribution 4.0 International License (CC-BY 4.0)

ABSTRACT

The escalating complexity of regulatory environments poses significant challenges to traditional compliance systems, which are inadequate for handling dynamic, unbounded tasks involving large volumes of unstructured and ambiguous data. We introduce an AI-based compliance verification framework leveraging Generative AI and Large Language Models (LLMs) to systematically address these issues. The framework comprises three components: (1) **Knowledge Acquisition**—extracting and structuring rules from unstructured texts into an ontology; (2) the **Verification Language Model (VER-LLM)—**applying logical reasoning and hypothesis testing to navigate rule ambiguities and assess compliance; and (3) **Continuous Improvement**—a feedback mechanism that refines the system's reasoning by identifying and resolving ambiguities in compliance requirements. The VER-LLM is trained on synthetically generated data derived from real compliance cases, ensuring privacy while capturing the nuanced patterns necessary for effective verification. Experimental results indicate that our framework outperforms the traditional LLM approach in accuracy, adaptability, and scalability, providing a real-time solution that adjusts to evolving regulations. This work significantly advances compliance verification, offering enterprises a sophisticated tool for accurate and efficient compliance enforcement.

Introduction

The growing complexity of regulatory environments, combined with vast amounts of unstructured and structured data, poses significant challenges for large enterprises—particularly in the financial, insurance, and legal sectors. Ensuring compliance with constantly evolving laws, policies, and industry regulations requires navigating intricate information embedded in contracts, policy manuals, and regulatory documents. These texts are often written in complex, formal language, making the interpretation of rules difficult and open to multiple readings. Failure to effectively manage compliance can lead to inefficiencies, errors, and substantial financial risks.

Traditional compliance systems have relied on rigid, rule-based approaches to manage regulatory obligations [1]. However, these systems are often ill-equipped to handle the dynamic nature of compliance, especially when dealing with complex datasets or cross-jurisdictional requirements. For example, adhering to intricate royalty payment structures or client-specific guidelines demands flexibility and accuracy that traditional systems cannot easily provide. Moreover, these systems struggle with the ambiguity inherent in legal and regulatory documents, as they rely on deterministic frameworks where inputs and outputs are clearly defined.

Compliance verification exemplifies an unbounded problem. In such tasks, not all variables or conditions are explicitly provided, requiring systems to work with incomplete information. This necessitates the generation and validation of multiple hypotheses to assess whether a particular rule applies in a specific case. The unbounded nature of the problem implies that the system must infer, reason, and iterate over assumptions,

filling in gaps where information is missing or unclear. As new information emerges, the system must adapt, adjusting its assumptions and re-evaluating its conclusions to ensure accurate compliance enforcement.

This combination of rule ambiguity and unbounded problem characteristics requires a flexible, adaptable solution that can handle the uncertainty and complexity of compliance tasks. This is where the power of Generative AI and Large Language Models (LLMs) comes into play. [2] LLMs have been deployed in increasingly complex, unstructured tasks across various industries. While they excel in generating text and interpreting structured data, their true potential lies in reasoning-intensive scenarios such as compliance verification. Unlike traditional systems, LLMs can navigate ambiguity, conflicting rules, and evolving regulations—making them an ideal tool for addressing the challenges of unbounded compliance tasks.

Our AI-based compliance verification solution integrates LLMs with advanced reasoning techniques, moving beyond the limitations of rigid, rule-based systems. By dynamically generating and validating hypotheses, accounting for ambiguous rules, and adapting to new information, this approach offers a scalable, real-time solution for complex compliance tasks. The system can extract rules, policies, and regulations from various sources—such as contracts, policy documents, client communications, and historical data—and enforce these across large datasets in real time. Whether it is modifying data based on complex payment structures, ensuring compliance with geographic regulatory variations, or adhering to client-specific guidelines, the system continuously adapts to evolving regulations.

By automatically identifying non-compliant items and suggesting corrective actions, our system minimizes risks and prevents financial losses. This proactive approach to compliance management addresses the core issue many enterprises face—getting “lost in the data”—and ensures a more efficient, accurate, and adaptive response to regulatory obligations.

Overview of the Framework

Our framework for compliance verification is designed around three key components: **knowledge acquisition**, **VER-LLM**, and **continuous improvement**. These components work together to manage rule ambiguity and handle the unbounded nature of compliance tasks, which require logical reasoning and hypothesis testing.

1. **Knowledge Acquisition**: This phase focuses on extracting rules from unstructured documents like contracts and guidelines, organizing them into a structured ontology, and ensuring they are available for application in specific cases.
2. **VER-LLM**: In this phase, the system applies logical reasoning to verify compliance. VER-LLM evaluates each case by testing different hypotheses, iterating through possible scenarios, and determining the correct reasoning paths to apply the relevant rules. The focus here is on ensuring accurate, structured decision-making without ambiguity.
3. **Continuous Improvement**: The system continuously learns and refines its reasoning process by tracking enforcement results and improving its handling of complex rules over time, making the verification process

more precise and adaptable.

This approach combines logic-driven hypothesis testing with real-time adaptability, allowing the system to deliver accurate, scalable compliance verification across dynamic environments.

Knowledge Acquisition

The **knowledge acquisition** phase forms the foundation of our compliance verification system. Given the complex and often unstructured nature of regulatory documents, contracts, and internal guidelines, this step focuses on accurately extracting and organizing the rules necessary for effective compliance checks. Since many of these documents are not part of the pre-trained data available to large language models (LLMs), integrating this domain-specific knowledge is essential.

One of the main challenges in compliance tasks is that the rules are often dispersed across various documents, expressed in complex grammatical forms, or supplemented by additional amendments. To address this, we employ a structured approach to extract, contextualize, and represent these rules:

1. **Contextual Rule Extraction**: Specific rules and their constraints are extracted from documents, ensuring that both general rules and their exceptions are identified. For example, a rule like “Travel expenses should not exceed USD 2,500” might have an exception for high-cost locations, which must be both captured and properly contextualized.
2. **Entity Linking**: After extracting the rules, we link relevant terms (e.g., "high-cost locations") to external knowledge bases to ensure accurate interpretation of the conditions under which rules apply.
3. **Ontology Integration**: Extracted rules and their exceptions are then organized into a hierarchical knowledge graph. This graph enables structured, logical retrieval and consistent application of all relevant rules, ensuring that even complex relationships between regulations are preserved.[3]

To maximize accuracy, we use a combination of open-source and proprietary models, fine-tuned for information extraction from legal and regulatory texts. Depending on privacy and performance requirements, the system can dynamically select the appropriate model for a specific task.

Our system’s flexible data structure (**Rules library**) ensures that the knowledge base is constantly updated to reflect changes in regulations or amendments to existing documents. This real-time adaptability is possible because there are links between rules and documents, as well as between documents themselves. When a document is updated, the system can automatically update all related rules, ensuring continuous compliance monitoring and reducing the risk of outdated or incomplete rule enforcement.

For storing rules, their embeddings, and metadata, we use the Deeplake storage [4].

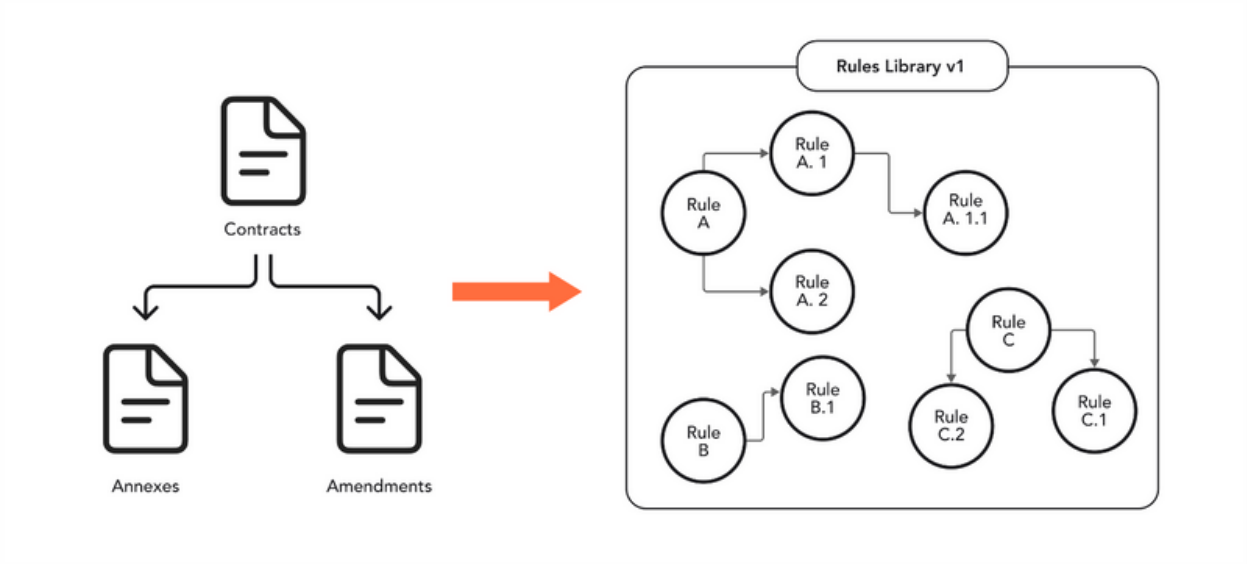


Figure 1. Ontology of the rules

Verification Language Models

When evaluating compliance, often no definitive answer or clear precedence exists, requiring clear logical reasoning to determine whether a specific rule or set of rules can be hypothetically violated in each case.

For such tasks, methods like Chain of Thought (CoT), Tree of Thought, and others are already popular [5]. However, none of these approaches generalize well to address the specific logic required for certain kinds of compliance checks. Models benefit from extracting distilled properties of an object and then using them to make decisions [6], rather than relying on general-purpose reasoning steps, such as in CoT. This kind of task-specific structure can significantly improve the quality of results.

To ensure these tailored reasoning processes are consistently applied, we follow a multi-step process to source, generate, and validate training data, which we can utilize to finetune verification expert models.

Data

To generate this training data, we source a large amount of real compliance data, both structured and unstructured, and reformat as well as classify that data based on expert definitions of the domain in question. This transformation is done utilizing existing foundation models, resulting in a large corpus of example data. However, this data by itself is unsuited for model training due to:

- Containing privacy-sensitive data that could be reproduced by models trained on it.

- Being unverified, i.e., lacking a ground truth on compliance enforcement.

Instead, we utilize a data generation process where each individual item is weighted based on its frequency and complexity. This complexity was previously judged during the transformation process, and a small, random subset of these items is provided to each individual request used to generate synthetic data.

The synthetic data generation model is then tasked to create novel items based on the real examples provided; however only does so given a series of constraints:

Most importantly, all private data found in the examples is anonymized and confirmed to be no longer present via further checks such as comparing synthetic results and real examples by applying Parts-of-Speech tagging and lemmatization to identify and prevent data-bleed.

Each set is also modified via further branches of conditions that can be combined to create near infinite variations for the resulting generations. Such conditions may focus the synthetic data generation model on particular sections within the example data to ensure each part of a verifiable item's language is attended to in the distribution of generated synthetic requirements or prompt the model to observe the examples under the perspective of an expert-annotated category of compliance requirements. We manually define input-data and prompts for each of these modifiers, which however are widely applicable to any kind of compliance data.

Rephrasing and modifying real data in such a way ensures that we approximate a more effective format for compliance-verification tasks [7], as well as create a more comprehensive distribution of different domain problems.

Based on these combinations of real seed-data and modifiers, we task the generation model to create both requirements that must be verified, as well as realistic cases that either satisfy or violate those requirements.

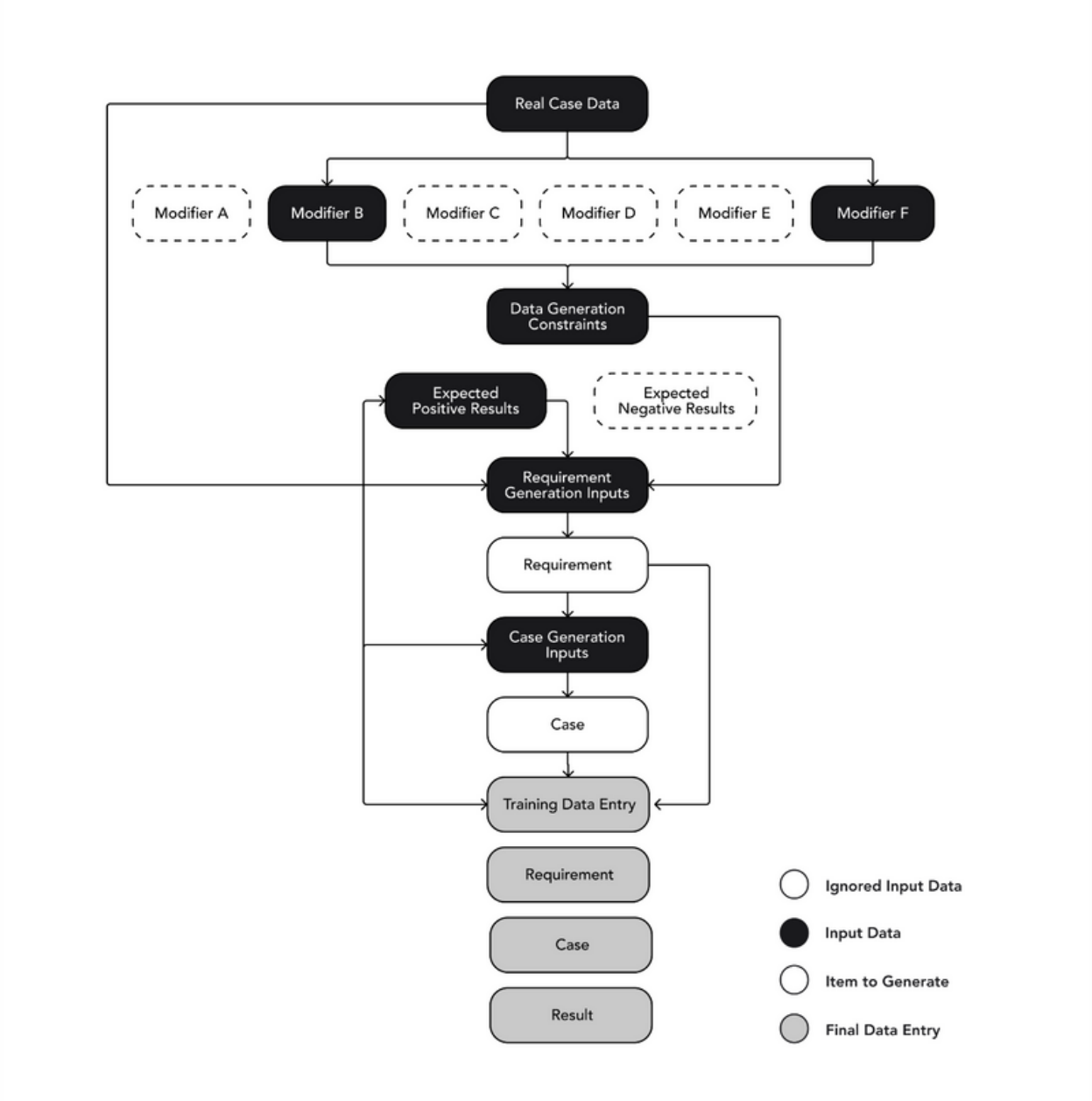


Figure 2. Synthetic data generation pipeline

One important concept in this generation process is the reversal of the verification task itself. Given exclusively real data, the only way to annotate compliance enforcement is to match a requirement to a case and form a conclusion on whether the requirement was adequately followed. However, since this is the exact task we wish to optimize our models for, generating synthetic data of these enforcements will necessarily result in a quality ceiling determined by the capabilities of the foundation model generating the data.

When reversing the process and generating requirements for a given case and expected enforcement result, or generating a case for a given requirement and expected enforcement result, foundation models are capable of

generating items with much higher confidence, as no potentially ambiguous judgement of two rigid items is to be generated, but instead a flexible data-point that matches a rigid expected result.

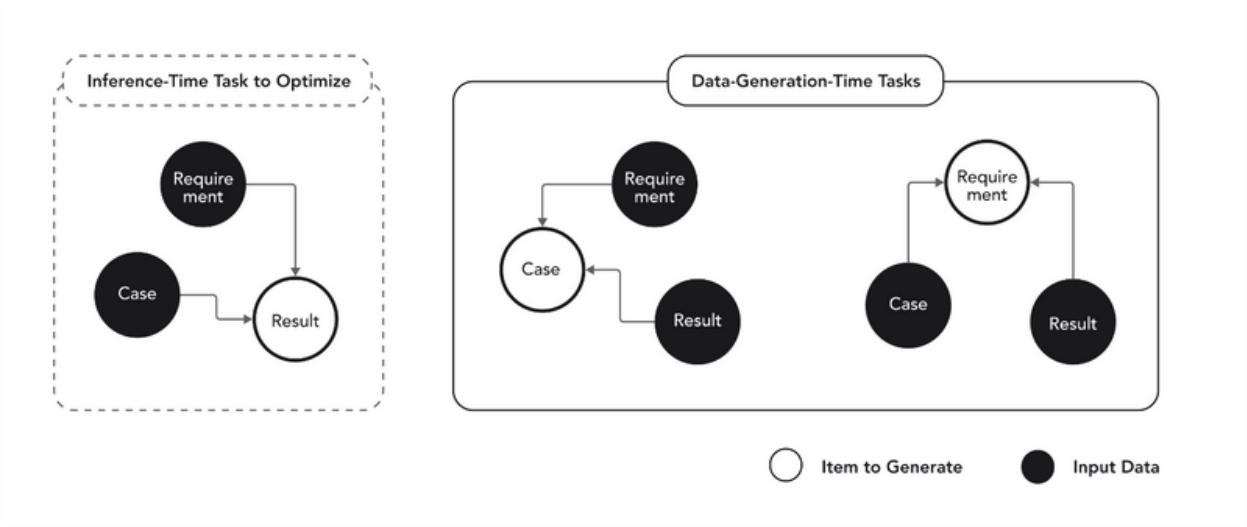


Figure 3. Augmentation of data generation process

As an output of this pipeline, we create highly diverse (due to the large amount of seed-data combined with varying modifiers) data-points, that each come pre-annotated with an expected verification outcome.

Based on this outcome, we can then prompt our generation model to attempt an actual enforcement given a pair consisting of a requirement and a case, and compare the resulting outcome against the expected result.

If it is not a match, we have identified a highly valuable property of the requirement/case pair: its complexity. If the requirement was straightforward to enforce, both models would likely succeed. However, if it is either very difficult to enforce or so ambiguous that there are multiple valid interpretations, the models may disagree.

Based on this insight, we can then persist the reasoning of the enforcement that resulted in the mismatch and feed it back to the enforcement model with additional instructions to re-evaluate it. In doing so we are not giving the enforcement model leading instructions; it is still free to choose whatever conclusion it wants to. However, as long as that conclusion does not match our expectation, we can keep feeding its previous reasoning back to itself to improve on it and enhance the overall reasoning process.

When the model eventually arrives at the correct, expected conclusion, we are left with an extensive reasoning trace that will crucially consider different viewpoints [8], eventually recognize initial reasoning errors, and pivot to the correct conclusion.

In practice, including these lengthy reasoning traces of complex requirement/case pairs in our training data teaches our models to not only be able to correct themselves instead of constraining themselves by their initial conclusions, but also to spend more tokens, and therefore more compute, on comparatively difficult

enforcements. This is variable test-time compute, a key component of modern reasoning-models (such as OpenAI's o1 series of models [9]), and due to our tailored pipeline for compliance task data-generation, we have a unique ability to synthetically arrive at and identify such complex cases for which models need to be taught to think longer before arriving at a final judgement.

In many ways, this mirrors human enforcement of such requirements: In complex cases, discussions between multiple conflicting viewpoints and interpretations are necessary to arrive at a fair result.

By iterating this process over sets of open and proprietary data, we managed to gather about 100k unique reasoning paths, which we later use to train our own models.

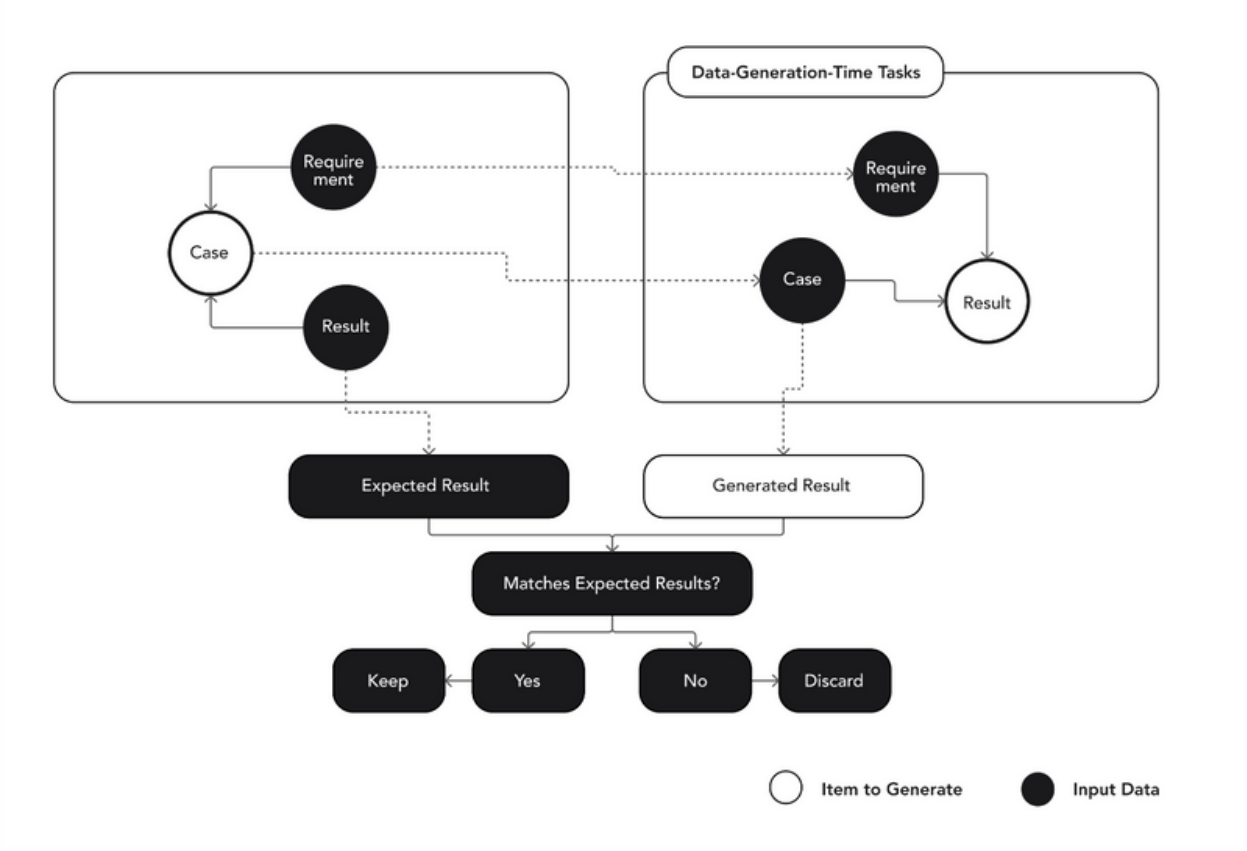


Figure 4. Process of filtering synthetic data to retain only valid entries

Training

We format the results of above data-generation process into individual verification entries consisting of:

- a requirement
- a "case" for the requirement to be verified against

- a verification result (compliant/non-compliant)
- detailed reasoning for why the generation model arrived at the result

For each generated reasoning, we validate that it follows our expected reasoning structure. We also amend each Requirement/Case pair of input data with additional annotations to aid the model with potential weaknesses related to tokenization or ambiguity in the data.

This annotation includes tagging grammatical constructs within the case-data that should be verified, as well as clarifications for the requirement-data (see “Continuous Improvement” section).

We finetune various open-source models on the resulting verification dataset, and achieve significant improvements on evaluations against real-world compliance verification tasks:

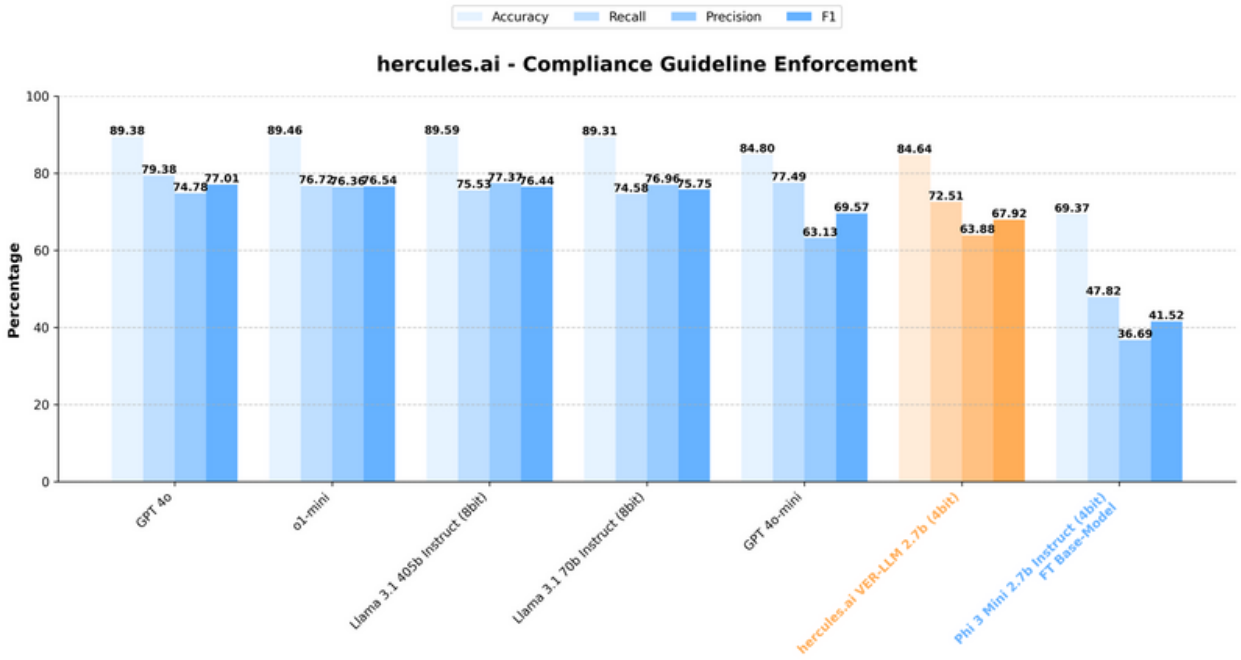


Figure 5. Enforcement quality metrics for finetuned and vanilla models

Continuous Improvement

Over the course of analyzing thousands of compliance documents and rules as well as their enforcement, one of the most critical challenges we identified has been the ambiguity inherent to much of the language used when formulating these requirements.

A human may write a perfectly coherent-looking requirement in a compliance document that uses well-formed language and appropriate terminology yet is often unable to anticipate all permutations of data this requirement

will apply to in practice. As a result, some of these permutations will expose ambiguities within the original compliance text: Should this policy apply to all forms of employee communication, including informal chats? Does “promptly” reporting an incident mean immediately or within a certain number of days? Which types of customer feedback are considered formal complaints under the compliance guidelines?

We find that any AI model that is attempting to enforce these requirements in isolation will act erratically. Unlike human enforcers, LLMs are trained to capture the full breadth of human knowledge, personalities, and therefore biases. These biases are brought to the surface by complex activation patterns, where a small change in phrasing within the input data can surface an entirely different interpretation of the same requirement.

Anticipating all these permutations over a wide area of possible requirements and subsequently trying to train a model to hold a consistent bias towards all of them is near-impossible; no training set could cover all of these cases in sufficient detail and number to induce consistent enforcement.

Instead, we train our models to, on each individual enforcement of a requirement over a data-point, identify any assumptions made during this enforcement. This process is kept entirely distinct from the actual enforcement action taken to ensure that a consistently neutral judge-model assesses each enforcement.

Assumptions describe any interpretations or novel conclusions the enforcement-model has made, and judge whether these are sufficiently captured by the definition of the original compliance requirement. If not, we generate a "clarification" that formalizes this weakness of the original definition. The clarification is persisted and given to all future enforcement iterations, therefore providing the enforcement-model with additional context generated by the judge-model. Using this additional context, the enforcement-model can now consistently apply the same assumption it made in a previous enforcement for all future cases where the same logic is necessary.

We also allow the judge-model to modify previously made clarifications. This effectively gives full autonomy to our judge-model to refine a compliance requirement over time, until it is sufficiently detailed. This serves as an agentic "scratchpad", acting as the memory of our models. The refinement process is limited, meaning we tune our judge-model to never exceed a certain level of complexity for generated clarifications. Instead, it will autonomously experiment with different phrasing and edits until the frequency of those edits approaches zero. This approach is similar to the one proposed in [10], however, we use a unified scratchpad with refinements.

This provides us a novel and effective measure of requirement-ambiguity.

Given a compliance requirement and a large set of enforceable data (which may be user-data or fully synthetic), we can rapidly execute both enforcement- and judge-models over that data to measure the time-until-convergence, i.e. the median number of enforcements necessary until we can expect no further interventions by the judge-model to be necessary. In other words, we measure the time until improvements on the requirement definition have saturated.

The resulting score can then represent the overall ambiguity of a requirement; a valuable insight into compliance texts that may feature hundreds or thousands of individual requirements.

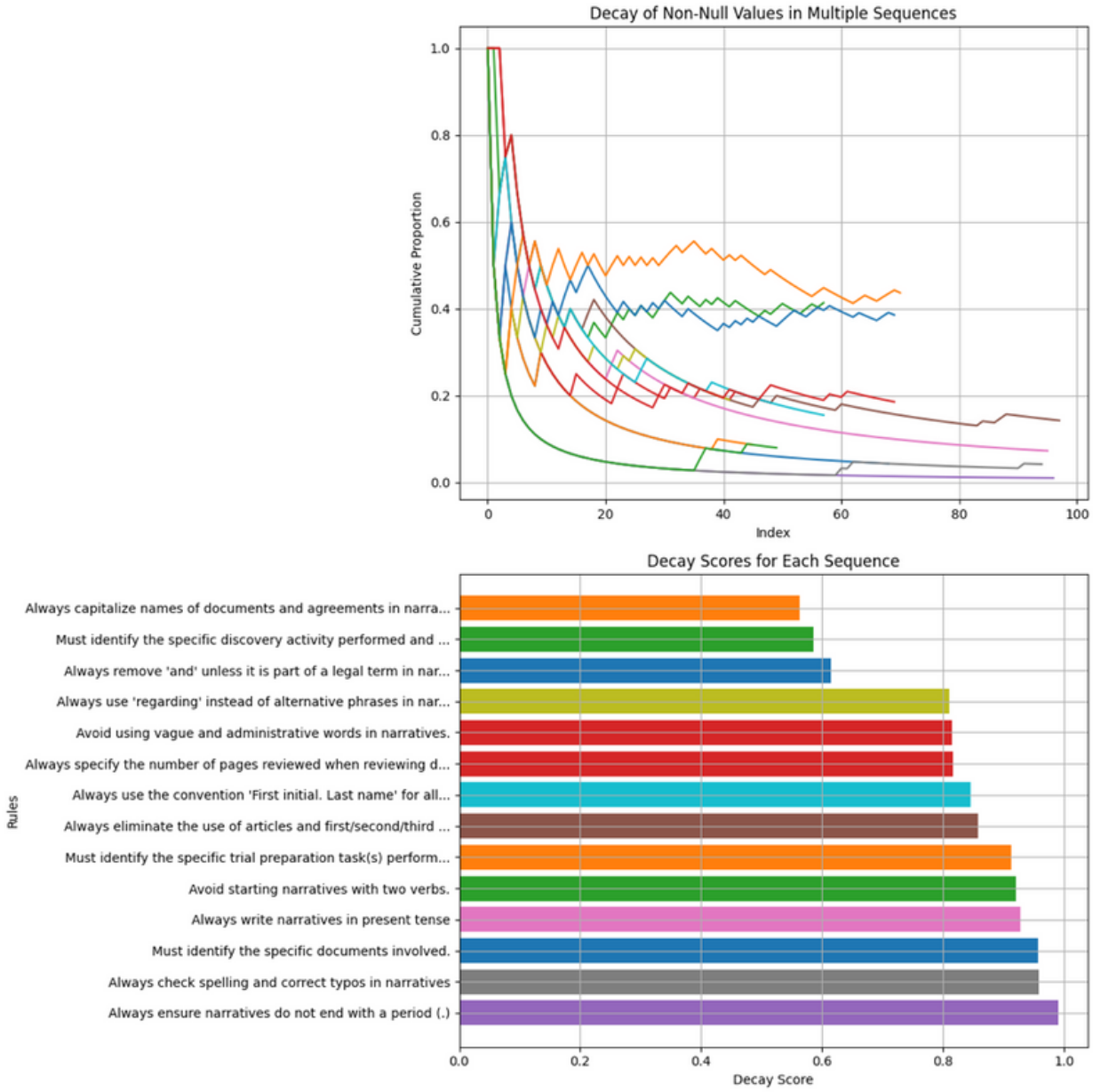


Figure 6. decay chart and ambiguity scores of different requirements

Three requirements do not converge over the given number of enforcements, resulting in a low score indicating significant ambiguity within the original text.

The resulting ambiguity scores can then allow us to surface problematic requirements to the end-user, asking for confirmation of the judge-model's conclusions or manual user-intervention overriding the original requirement text. Our enforcement model is able to respect either one the same.

Generating clarifications in concise, natural language instead of more abstract constructs such as decision trees also allows humans to easily review and edit these clarifications in the same way they would clarify a requirement to a co-worker or in internal documents, while directly impacting the model behavior in a transparent way.

Following this process also allows us to measure a gap in enforcement-consistency if the enforcement-model is provided with clarifications from the judge-model:

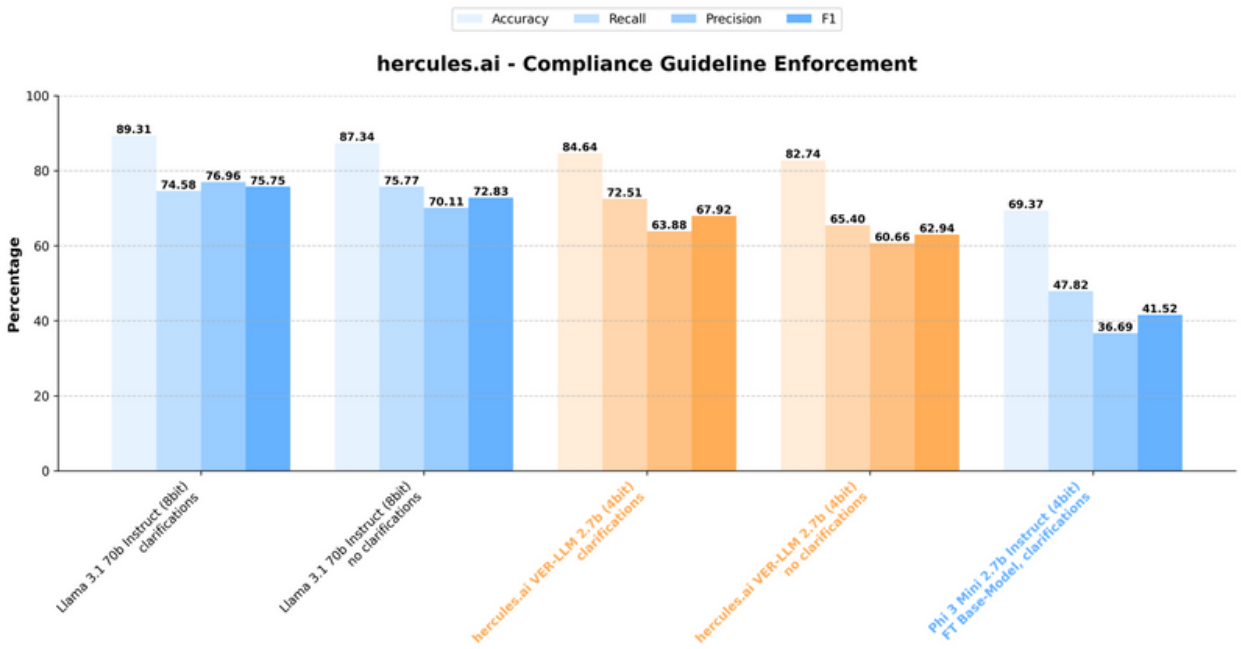


Figure 7. Performance gap before and after introducing clarifications

Conclusion

The framework presented in this paper demonstrates a novel approach to handling the complexities of compliance verification. By combining large language models (LLMs) with advanced reasoning techniques, the system can manage the ambiguity of rules, handling unbounded tasks, and iterating through hypotheses to find the correct application of compliance rules.

The key components of the system—**knowledge acquisition**, **VER-LLM**, and the **continuous improvement** process—work together to provide a flexible, scalable solution that can adapt to evolving regulations and unstructured data. The knowledge acquisition process ensures that rules are accurately extracted and represented, while VER-LLM follows structured reasoning paths to verify compliance. The continuous improvement mechanism allows the system to learn from each case and refine its decision-making process over time.

This approach enables real-time compliance verification, offering clear, logical explanations for each decision, while also continuously improving as it processes more cases. The system's adaptability and precision make it well-suited for a wide range of industries that rely on dynamic compliance requirements.

By combining advanced AI techniques with practical applications, this framework represents a step forward in automating complex compliance tasks and ensuring accurate enforcement in environments where rules are often ambiguous or subject to change.

Limitations and Future Work

1. **Use of Small Models in Experiments**: In our experiments, we utilized smaller models (e.g., 2.7 billion parameters as a baseline) because, in real-world applications, we need to perform a large number of enforcements and iterate over hypotheses. Employing larger models would be computationally expensive and impractical for such tasks. Despite using smaller models, we achieved acceptable levels of accuracy, making them a viable option for large-scale compliance verification.

2. **Limited Contextual Information**: Our current approach focuses on enforcement using guidelines and available documents such as contracts. However, there is a substantial amount of additional information involved in the processes that specialists use when assessing compliance risks. This includes client agreements, generally accepted practices, historical patterns, and more. We plan to expand the domain context by incorporating this additional information to enhance the system's quality and effectiveness.

References

1. A. Elgammal, O. Turetken, W.-J. van den Heuvel, and M. Papazoglou, "Formalizing and Applying Compliance Patterns for Business Process Compliance," Software & Systems Modeling, vol. 15, no. 1, pp. 1–40, Feb. 2014. [Online]. Available: https://link.springer.com/article/10.1007/s10270-014-0395-3.
2. Hassani, Shabnam, et al. "Rethinking Legal Compliance Automation: Opportunities with Large Language Models." arXiv preprint arXiv:2404.14356 (2024).
3. Santos, Sarah, et al. "Requirements Satisfiability with In-Context Learning." arXiv preprint arXiv:2404.12576 (2024).
4. S. Hambardzumyan, A. Tuli, L. Ghukasyan, F. Rahman, H. Topchyan, D. Isayan, M. Harutyunyan, T. Hakobyan, I. Stranic, and D. Buniatyan, "Deep Lake: a Lakehouse for Deep Learning" Proceedings of CIDR (2023). Available at: https://www.cidrdb.org/cidr2023/papers/p69-buniatyan.pdf.
5. A. Kalyanpur, K. K. Saravanakumar, V. Barres, C. McFate, L. Moon, N. Seifu, M. Eremeev, J. Barrera, A. Bautista-Castillo, E. Brown, and D. Ferrucci, "Multi-step Inference over Unstructured Data," Elemental Cognition Inc. [Online]. Available: https://arxiv.org/pdf/2406.17987#page=10.06. Accessed: https://ec.ai/research-papers/.

6. P. Zhou et al., SELF-DISCOVER: Large Language Models Self-Compose Reasoning Structures, pp. 1–17, Feb. 2024. [Online]. Available: Available: https://arxiv.org/abs/2402.03620.
7. Pratyush Maini et al., Rephrasing the Web: A Recipe for Compute and Data-Efficient Language Modeling. [Online]. Available: https://arxiv.org/pdf/2401.16380
8. M. Hu, P. Zhao, C. Xu, Q. Sun, J. Lou, Q. Lin, P. Luo, S. Rajmohan, and D. Zhang, "AGENTGEN: Enhancing Planning Abilities for Large Language Model based Agent via Environment and Task Generation," The University of Hong Kong, Microsoft Corporation. [Online]. Available: https://arxiv.org/pdf/2408.00764.
9. OpenAI, "Learning to Reason with LLMs," OpenAI. [Online]. Available: https://openai.com/index/learning-to-reason-with-llms [Accessed: Oct. 23, 2024].
10. H. Sun, Y. Jiang, B. Wang, Y. Hou, Y. Zhang, P. Xie, and F. Huang, "Retrieved In-Context Principles from Previous Mistakes," Peking University, Beijing Institute of Technology, Alibaba Group, Chinese Academy of Sciences. [Online]. Available: https://arxiv.org/pdf/2407.05682.
11. B. Chen, C. Shu, E. Shareghi, N. Collier, K. Narasimhan, and S. Yao, "FIREACT: Toward Language Agent Fine-tuning," System2 Research, University of Cambridge, Monash University, PLI, Princeton University. [Online]. Available: https://arxiv.org/pdf/2310.05915.
12. S. Qiao, N. Zhang, R. Fang, Y. Luo, W. Zhou, Y. E. Jiang, C. Lv, and H. Chen, "AUTOACT: Automatic Agent Learning from Scratch for QA via Self-Planning," Zhejiang University, Zhejiang University - Ant Group Joint Laboratory of Knowledge Graph, AIWaves Inc., Alibaba Group. [Online]. Available: https://arxiv.org/pdf/2401.05268.
13. M. Hu, P. Zhao, C. Xu, Q. Sun, J. Lou, Q. Lin, P. Luo, S. Rajmohan, and D. Zhang, "AGENTGEN: Enhancing Planning Abilities for Large Language Model based Agent via Environment and Task Generation," The University of Hong Kong, Microsoft Corporation. [Online]. Available: https://arxiv.org/pdf/2408.00764.
14. L. Garza, L. Elluri, A. Kotal, A. Piplai, D. Gupta, and A. Joshi, "PrivComp-KG: Leveraging Knowledge Graph and Large Language Models for Privacy Policy Compliance Verification," University of Texas at El Paso, Texas A&M University Central Texas, University of Maryland Baltimore County. [Online]. Available: https://arxiv.org/pdf/2404.19744.
15. R. Padhye, "AI-Driven Statutory Reasoning via Software Engineering Methods," Sept. 11, 2024. [Online]. Available: https://law.mit.edu/pub/aidrivenstatutoryreasoning/release/1.