THE
CARTER CENTER

Technical Report on Using LLMs for Legal Coding

—A Freedom of Expression Report—

February 16, 2026

Raúl Alejandro Pérez Saucedo, Emily Adams, Hans-Inge Langø,
Anthony J. DeMattee

Democracy Program, The Carter Center

How to Cite This Report

For citation purposes, please use the following format:

Raúl Alejandro Pérez Saucedo, Emily Adams, Hans-Inge Langø, Anthony J. DeMattee. 2026. "Technical Report on Using LLMs for Legal Coding." The Democracy Program, The Carter Center. Atlanta, GA.

The Carter Center
453 John Lewis Freedom Parkway NE
Atlanta, GA 30307-1406

`www.cartercenter.org`
General Inquiries: +1 (404) 420-5100
Donor Services: +1 (404) 420-5109

Executive Summary

We explore the possibilities of using large language models (LLM) to parse complex legal texts in support of a project collecting data on expression, defamation, and mis-/disinformation laws globally. Training and employing humans to label laws is costly, and building a global corpus requires extensive language skills. To assess whether we can automate this process, we develop and test two competing approaches: one using high-level, off-the-shelf GPT reasoning models, GPT-4o and GPT-5, and one hybrid approach using a combination of GPT models and a fine-tuned BERT model for labeling. We find that the two-stage GPT-5 configuration performs best, correctly identifying 49% of exact institutional statements and 65% of topics. While these results are insufficient for fully automated labeling, the system is able to surface relevant provisions and structure them to reduce burden on human coders. We conclude that current LLMs work best as a supplement rather than a replacement for human labelers. We offer several lessons learned and next steps for developing LLM-assisted labeling.

Contents

1 Introduction

We explore the usefulness of LLMs in parsing complex legal text. Building on an existing project that attempts to understand *de jure* freedom of expression in different countries across the world, we combine AI-assisted procedures with Institutional Grammar frameworks (Crawford and Ostrom, 1995; Ostrom, 2005) to label laws at the provision level. We test several approaches, ranging in levels of complexity and cost, to assess whether these technologies can assist or replace human labelers.

Using humans to label laws is a time- and resource-intensive process as it requires significant upfront costs to train individuals and compensate them for their time reading and analyzing the texts. Additionally, there is sometimes disagreement between human coders on what is and is not in the scope of the codebook. Finally, different languages present a barrier to human coders, reducing the scope and representativeness of the legal corpus. To address these issues, we test integrating LLMs into the labeling process.

Large language models (LLM) can parse dense text, perform high-level reasoning tasks, and generate outputs. Because of how legal texts are structured, with clear syntax and rule-based approach, the legal domain is a particularly promising use case for this technology. AI-assisted labeling is a developing field of legal research, with recent advancements in LLMs showing higher accuracy and lower hallucination rates (Dragoni et al., 2016; Janatian et al., 2023; Westermann and Benyekhlef, 2023). However, performance depends on a wide range of factors, including the choice of models and levels of customization.

We test two general approaches to integrating LLMs into the labeling process. The first approach uses an off-the-shelf GPT model and creates specification through multiple stages and prompt engineering. The second approach utilizes a GPT model to parse through the legal text to identify and separate relevant text snippets, then uses a fine-tuned legalBERT model to classify each legal rule with the corresponding institutional statement. We analyze the performance of these two approaches against the human labeling and against each other to understand how well each performs.

We find that the GPT-5 models perform the best, followed by the legalBERT, with GPT-4o models performing the worst. The best performing model is able to correctly identify the exact institutional statement 49% of the time, and the correct topic 65% of the time. Based on this, we show there is promise for integrating LLMs into legal research and text parsing tasks without compromising the validity of the results. However, it is not suitable as a substitute for human labeling, as the models do not achieve sufficiently high coverage. Additionally, we find that a two step LLM approach is able to bridge the precision-recall tradeoff by maximizing for each in separate steps.

This report proceeds as follows: first, we describe the sample and problem that this is testing. Then we describe the methodology and the specifications and trade-offs of the differing approaches. Then, we discuss the results and finally, draw conclusions and point to next steps in furthering this specific project and extensions to the research.

2 State of the Field

Laws are a promising subject for LLMs because of their natural language processing capabilities. However, the idea of integrating artificial intelligence into legal research is not new. The study of applying AI to law goes back decades, with early scholars emphasizing the importance of argumentation in law as a key requirement for evaluating the abilities of AI approaches (Bench-Capon, 1997; Bench-Capon et al., 2012). With the issues of hallucination in models today, we integrate this foundational principle into our approach.

One strategy of legal research views legal texts not as purely narrative language, but as items that can be analyzed as structured, codable rule systems (Ostrom, 1990; Ashley, 2017). This domain feature makes laws an ideal subject for systematic computational analysis. We utilize Institutional Grammar framework introduced by Crawford and Ostrom (1995) to create this structure (described in detail below).

One of the key developments that has accelerated these efforts in recent years has been the digitization of legal texts, which in turn has aided the development of early analytics tools (Frankenreiter and Livermore, 2020). More recently, the development of LLMs presents an opportunity for scaling legal analysis and research(Frankenreiter and Livermore, 2025). While there are limitations to current models, including hallucinations, accuracy, bias, and ethical responsibility, the opportunity for advancing research means that LLMs provide a very promising expansion of current work.

One of the main features of LLMs is that they can be trained or fine-tuned for specific tasks or domains. However, there is no clear agreement on the relative performance of specialized versus general models at performing legal tasks. While specialized models tend to perform better at some tasks, general models still perform better at others, leaving the question of whether it is worth developing specialized models up in the air (Guo et al., 2025; Savelka and Ashley, 2023). Prior work in legal NLP consistently finds that transformer encoders such as BERT excel at localized classification and span detection tasks, while higher-level reasoning and rule synthesis benefit from more expressive models (Savelka and Ashley, 2021). Taken together, we test the performance of GPT versus BERT models at our legal task, comparing a high performing off-the-shelf model, with a fine-tuned specialized model.

Recent work suggests that effective use of LLMs for complex text analysis depends on workflow designs that decompose tasks into steps. Specifically, a verification step where outputs are checked against source texts before being accepted have shown to reduce hallucinations, therefore improving reliability (Dhuliawala et al., 2024). Moreover, separating the identification of candidate provisions from subsequent verification means there is a higher focus on recall early, while precision is improved later in the process with text-grounded verification (Janatian et al., 2023). Because of this, we separate the evidence and justification outputs from the LLM, allowing legal coders to assess both the statutory basis and the reasoning behind each decision.

3 Sample

In order to train and test our models, we use a set of laws that have been coded by human labelers for a larger researcher project. The sample consists of 164 laws pulled from 62 countries and 4 international organizations from 1894 to 2025. These laws all contain legal rules that govern expression in their respective regions. From this corpus, 25 laws were retained as a test set to assess the performance of each approach while the rest were used in the fine-tuning of the BERT model.

We use the human-generated labels as "ground truth" against which to compare the results of the LLM based classification. Human-to-human inter-coder reliability is above 90% in the human-coded sample of laws, meaning that while there is some variation, it is largely consistent and reliable.

The laws in our legal corpus vary widely along many variables including length, frequency of items coded, and topics covered. In order to create a representative test sample, we grouped them along similar characteristics to approximate these variations. The test sample was selected using a stratified random sampling method to increase the representativeness of the sample. We grouped laws into categories: Emergency, Penal and Criminal Codes, Cybersecurity, Press, Digital Age, and a General Category. Random samples were taken from each group, with the sample size proportionate to the total coded legal corpus to make up a sample set of 25 laws.

Using a global legal corpus presents certain challenges to creating a unified framework for labeling the laws. They are written in many different languages and exist within legal frameworks that vary by country and legal tradition. Recent advancements have improved the ability to translate laws. To some extent, LLMs have also been able to parse and code laws in major languages. We see the models perform well on English, Spanish, and French laws without translation. Because of the varying legal contexts (e.g., different languages and

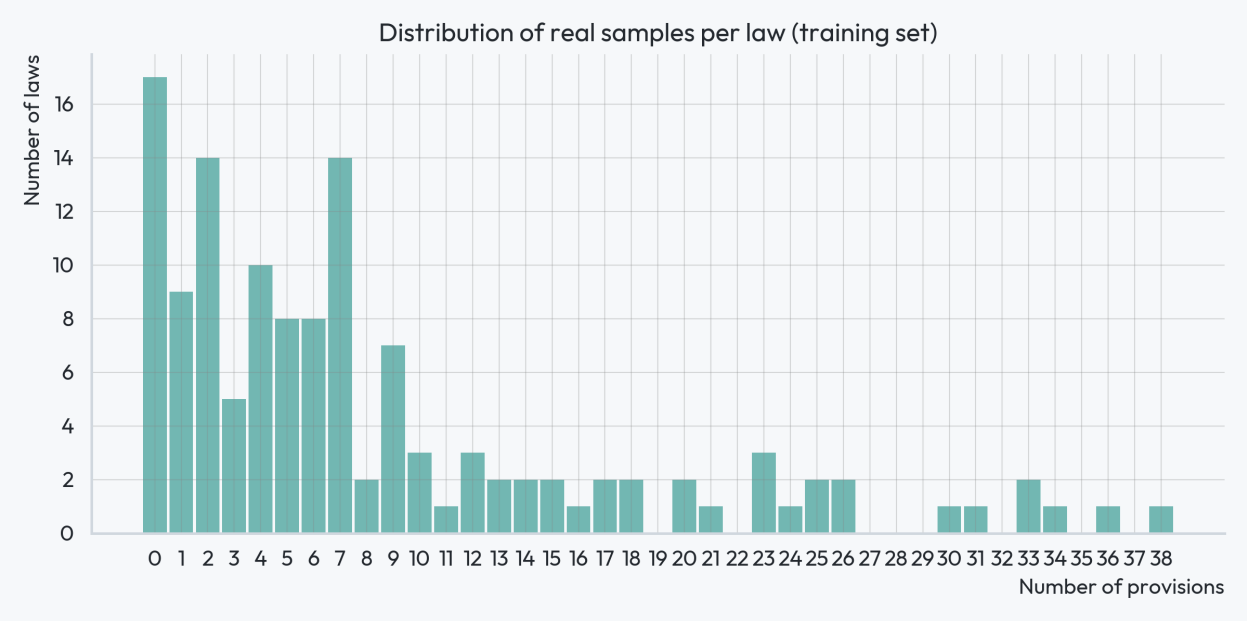


Figure 1: Most laws have fewer than 8 provisions, while some laws have more than 30.

legal traditions), automating the parsing of laws has previously not been predictable, as the different structures make it difficult to reliably split laws into legal rules. LLMs present an opportunity to overcome this barrier because of their prowess for pattern recognition. This legal research is a slow, manual process, but because of the specific abilities of LLMs, they have the potential to streamline parts of the labeling process.

4 Methodology

Our goal for this project is to achieve reliable automated labeling of laws that can meet or exceed human performance. We are concerned with both precision and recall,[1] though recall is particularly important since it is easier to identify false positives than false negatives through human-led validation. However, we also have to be mindful of costs. State-of-the-art LLMs can perform high-level reasoning tasks with large context windows, but they are expensive to use, requiring cloud computing resources with high token counts, and expensive to fine-tune for our purposes. Because of this potential trade-off between cost and performance, we develop two competing approaches to automated labeling that have different cost and complexity profiles.

[1]Precision and recall measure different kinds of performance for a binary classifier. *Precision* focuses on the quality of positive predictions — e.g., if the classifier labels a law as having a certain coding item, is it usually right? By contrast, *recall* emphasizes the coverage of actual positives — e.g., did the classifier match the same coding items?

First, we develop a pipeline that uses different GPT models from OpenAI to perform zero-shot inference[2] from our law texts. We iterate over three versions, experimenting with different ways of pre-reading or validating the labeling output in order to identify the best-performing approach, without having to resort to fine-tuning.

Second, we build a pipeline that incorporates cheaper models with fine-tuning but requires more data preparation and careful design decision. The front part of this pipeline involves reconstructing training data from our human-labeled data in order to fine-tune a specialized BERT model. The back part splits up law texts using a smaller GPT model before passing the resulting text snippets to the fine-tuned model. All parts of the hybrid pipeline can be feasibly run locally with API calls, though we perform parts of it on the Azure Machine Learning portal to speed up fine-tuning and inference.

4.1 Law-Labeling Process

We use the Laws of Expression (LEX) dataset from Adams, Xu and DeMattee (2026) to measure the *de jure* expression environment, which catalogs and measures the legal rules that regulate freedom of expression. Using institutional grammar (Crawford and Ostrom, 1995; Ostrom, 2005), LEX systematically codes each law using a 722-item coding protocol grounded in international human rights standards. The protocol accounts for different actors, types of expression, and topics. LEX (2026, pp. 9-15) takes a two-step approach modified from DeMattee (2023*b*) to create country-year measures and several different *de jure* freedom of expression indexes ranging from normatively undemocratic to normatively democratic. We use the LEX dataset to compare legal institutions across countries over time.

We describe this methodology in further detail below, outlining the coding protocol. More information about the LEX dataset and methodology can be found in the codebook (Adams, Xu and DeMattee, 2026).

We start the law labeling process by identifying relevant laws, and then using online and library resources to build a legal corpus of the PDFs of the laws. Next, following a similar approach used to analyze the politics of laws restricting associational rights (DeMattee, 2019, 2020, 2023*a*) we code these primary documents using institutional grammar (IG) Crawford and Ostrom (1995); Ostrom (2005) to increase the comparability of laws across countries and time.

Known in institutional analysis as the ADICO syntax, IG uses five components to identify whom the institutional statement applies (*A*ttribute), denotes expected behavior (*D*eontic), prescribes particular action (a*I*m), specifies the circumstances under which it

[2] *Zero-shot inference* is instructing a model to make a prediction — in this case, label legal text — on new, unseen data without any specific training for this task.

applies (*C*ondition), and provides the institutionally assigned sanction for noncompliance (*O*r else). The value of of IG for comparative legal analysis and research is its ability to transform natural legal language into an institutional statement and facilitate comparison to similar institutional statements in use in a variety of contexts. By enabling rigorous, cross-country, over-time comparisons of legal texts, IG goes far beyond simple legal summaries—empowering researchers and analysts to systematically code legal rules and make sharp, data-driven comparisons across legal institutions. The text box below uses a Singaporean statute to illustrate taking legal rule from a law and converting it to an institutional statement from our codebook.

Example: Singapore 2019 Protection from Online Falsehoods and Manipulation Act

> **Text:** 7. (1) A person must not do any act in or outside Singapore in order to communicate in Singapore a statement knowing or having reason to believe that (a) it is a false statement of fact; and (b) the communication of the statement in Singapore is likely to (i) be prejudicial to the security of Singapore or any part of Singapore;
>
> **ADICO:** [Citizens][may not][exercise (or automate) disinformation that threatens national security][at all times and under all conditions][or else institutional sanction]

Institutional grammar outlines three types of institutional statements: rules, norms, and strategies. Our codebook includes both rules and norms. If all five components are present, the institutional statement is a rule. If a statement is missing the "or else," then there are no institutionally assigned consequences for breaking the rule, which makes the institutional statement a norm. For our project, this means that the fit to ADICO format cannot be too strict, or institutional statements that establish norms will missed.

For each institutional statement, the deontic consists of three operators: *permitted/may*, *forbidden/must not*, *obliged/must*. The deontic plays a crucial role, because of its ability to be flipped from permissive (may, must) to restrictive (must not) without changing any other context, creating a clear indicator of whether a provision expands or constrains expression.

Our codebook contains 722 institutional statements covering different combinations of actors, types of speech, topics, and conditions under which these laws arise and apply. This codebook is built on the basis of freedom of speech as defined by Article 19 of the ICCPR which establishes universal freedom of expression, but allows states to restrict expression to respect the rights and reputations of others or for the protection of national security, public

order, public health and public morals.

4.2 Evaluation Approach

IG Component	Definition	How Its Used In Our Work	Example Keys
Attribute (A)	Attributes to whom the institutional statement applies.	Encoded as the `Actor` field in the rule key (Necessary).	`C` (Citizen) `P` (Press) `G` (Government)
Deontic (D)	Operator specifying obligation, permission, or prohibition .	Encoded numerically: `+1`, `-1`, `0` (Necessary).	`+1`

Aim (I)	Aim describes particular actions or outcomes to which the deontic is assigned.	**Type** denotes the expressive act category (Auxiliary).
		Topic denotes the subject matter of the statements (Necessary).
			`EXPRESS`
			`DISINFO`
			
			`HATE`
			`ELECTION`
			`DEPLATFORM`

Condition (C)	Conditional qualifier under which the rule applies.	Conditional field in the rule key (e.g., crisis status, digital domain) (Auxiliary).	`CRISIS` `DIGI`
Or else (O)	Defines the sanctions imposed for non-compliance.	Specific consequences not collected in this work.	_____

Table 1: Mapping institutional grammar components to structured rule keys.

When comparing the performance of the different models, we rely on their precision, recall, and F1 scores. Because our is to integrate the use of the LLM into the human labeling process, we prioritize recall performance in sorting the outputs of the table. Maximizing recall increases the performance of the model on the most labor intensive part of the labeling process, parsing the law and identifying relevant sections.

We also evaluate the model at varying levels of specificity. By breaking down the institutional statements into their sub-components and looking beyond the exact match, we can evaluate the strengths and weaknesses of each model for differing goals. Figure 2 demonstrates the varying levels of specificity we employ in our analysis. We can therefore measure performance beyond the goal of replacing human labeling, where we would look for high performance on exactly matching the labeling items, and instead imagine the LLM as sup-

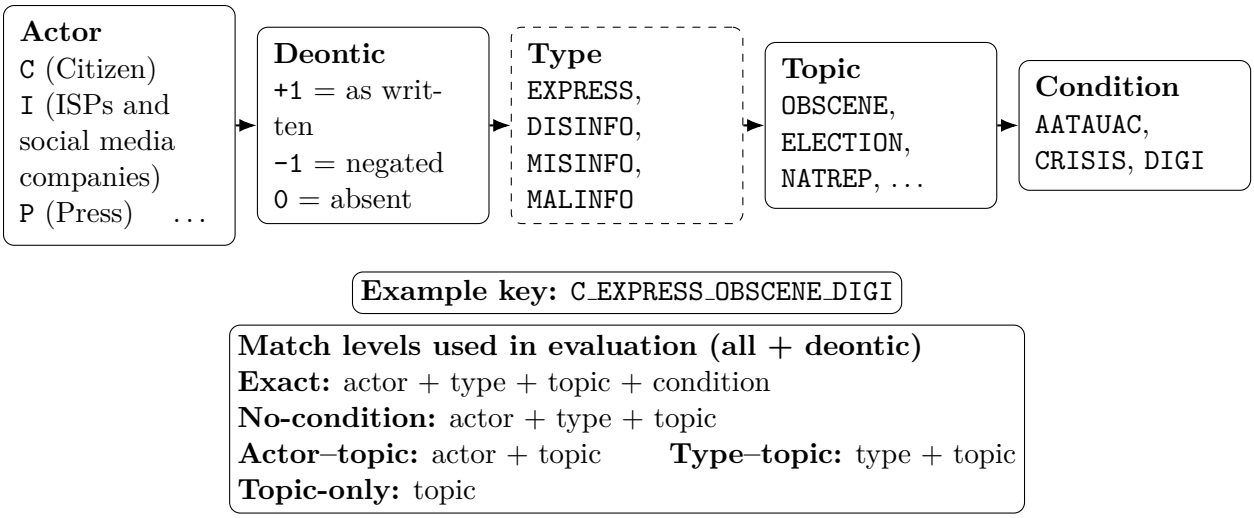


Figure 2: Solid lines denote components that are present in every key. Dotted boxes denote components that are not always present.

plementing human labeling by alleviating parts of the labeling process. When the model is able to identify the relevant sections reliably, the human can much more quickly go through the identified items and code the exact institutional statements based on the pulled text snippets.

4.3 Two-stage LLM Classification

For our first attempt at integrating LLMs into law labeling, we developed a two-stage pipeline using off-the-shelf LLMs. The LLM first receives the full legal text and labeling instructions, and we prompt it to return all provisions identified as negated (-1) or present (1) in the law, along with a verbatim citation (evidence) and an explanation (justification) for its labeling decision.

In the second stage, the law is first embedded into chunks. For each provision identified as negated or present in the previous stage, the model is prompted to verify the first stage classification, with further detail on the provision's full ADICO template. Instead of the full legal text, the model receives only the top-ranked chunks by semantic similarity. The output is also required to have verbatim evidence and justification.

This means that for each law, the total number of model calls equals one initial call for Stage 1 plus one additional call for each non-zero provision identified in Stage 1.

Model choice

The models we use for testing are GPT-4o and GPT-5 from OpenAI, accessed through their API. We use the same snapshot of each model across all versions of the pipeline to control for intra-model consistency. Both models perform strongly on legal reasoning benchmarks for LLMs (Guha et al., 2023), and both are suitable to handle large legal texts. GPT-4o supports a context window of 128,000 tokens, while GPT-5 supports 400,000 tokens. GPT-5 has a higher context capacity and greater capability for complex, instruction-heavy prompts, while GPT-4o is an earlier-generation model with a smaller context window and lower latency and cost.

Early iterations

We iterate over different prompt and data structures as we assess the citations and justifications from the model's output.

Version 1: Single-stage with full law and codebook. Most similar to human labeling, the model receives the full law and codebook and is prompted to return a single JSON output. Keys are actor-specific (citizens, press, platforms), so the model has to determine both whether a rule existed and which actor it applies to. Because of context-window limits, ADICO templates are excluded from the codebook, and classification relies mainly on topic similarity rather than structured IG labeling.

Version 1.5: Two-stage with previous *reading* step. We add a first-pass round that extracts metadata from the law and produces a structured summary of the law (domains and actors) to guide labeling that was included in the main prompt. We implement this part as a separate LLM call (`read_law`). We see no incremental gains in either labeling precision or justification clarity, so we skip this reading step in the next version.

Current version

Version 2: Two-stage with additional *verify* step Instead of a separate reading step, we add a second stage (`verify_law`) after first-pass labeling that checks each coded item against retrieved statute excerpts. The full law is first chunked and embedded; then, for each coded item, a separate LLM call receives its relevant chunks and is asked to verify the initial labeling, returning an updated label with quoted evidence and justification. Because `verify_law` makes one model call per non-zero provision from the first step, we can include the full ADICO template and the complete codebook in the prompt.

Pipeline design

We designed the evidence/justification setup to work with human oversight so that it is reproducible for other legal labeling frameworks. The pipeline is not designed to be an autonomous coder, but a tool whose outputs must be constrained and verifiable, where each design choice responds to a specific risk in the use of LLMs for legal labeling. The pipeline addresses common challenges in applying language models to legal labeling by combining constraints and structure.

To prevent over-inference, we instruct the model to justify its outputs solely based on the cited text, avoiding unsupported generalizations. Then, a two-stage setup balances recall and precision: the first stage casts a wide net to capture potential legal rules, while the second stage applies Institutional Grammar to eliminate false positives. To manage prompt saturation [3], the full legal text is provided only once initially, with subsequent steps using targeted excerpts retrieved through a retrieval-augmented generation (RAG) approach. Finally, by specifying the desired output format clearly, the pipeline ensures reliable, structured responses without overloading the prompt or increasing hallucination risk.

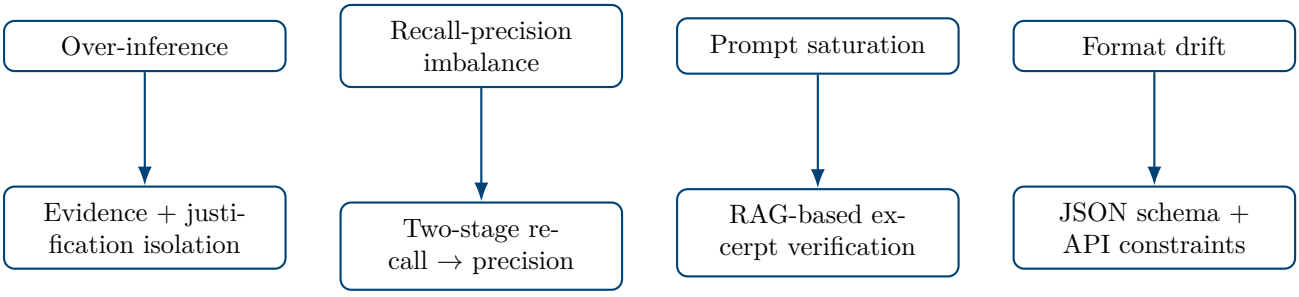


Figure 3: LLM risks in legal labeling to pipeline design choices that mitigate them.

- **Over-inference**: Language models tend to generalize from contextual clues to return confident answers. This is a natural part of their training, but in legal labeling, it can lead to inference of legal rules that humans wouldn't consider present. To guard against this, we force the LLM to justify its labeling based directly on the cited excerpt, thereby isolating the cited text from the model's generative answer and grounding it in the legal text.
- **Precision/recall imbalance**: Prompt engineering strongly influences LLMs. Minor prompt differences can lead the same model to produce very different outputs, making

[3] "Prompt Saturation" occurs when an AI Agent, gets overwhelmed by having to choose from too many actions, impairing its decision-making accuracy and efficacy.

it hard to both follow strict classification rules and capture all relevant provisions. A two-stage setup resolves this. The first, more permissive round minimizes missed legal rules, especially in long texts. The second round applies Institutional Grammar for structured classification that removes false positives.

- **Prompt saturation**: Recent state-of-the-art LLMs offer large context windows, often above 100,000 tokens. This, however, doesn't mean models perform the same with a 10-token prompt as with a 100,000-token prompt. Simple tasks like summarization can function well with large prompts, but logic-heavy reasoning, such as legal labeling, can saturate a single call and degrade model performance. The pipeline takes advantage of the large and growing context windows but also guards against its risks. To address this, the first stage of the pipeline receives the provided law in its entirety, once, while the second stage receives exclusively an excerpt of the law through a retrieval-augmented generation (RAG) method for logic-heavy, IG-based labeling, where only the most relevant text for each provision is necessary.
- **Structured outputs and instruction following**: Closely linked to prompt saturation, prompting for unconstrained outputs loads another decision onto the model and increases the risk of hallucinations. Classic prompt engineering techniques for older models recommended repeating instructions multiple times inside a single prompt. However, instruction-following is one of the aspects of LLMs that has greatly improved since the first models became widely available. Thus, specifying the required output format once in the prompt and constraining it through the API has proved sufficient for this pipeline, thus reducing silent failures such as skipped or partially answered items.

Prompt development

Legal labeling is not exempt from classic prompt engineering techniques to improve the performance of LLMs for this task. The following are applied in this project:

- **Role/persona definition**: "You are a legal expert trained in freedom of expression."
- **Explicit task specification**: "You will code a legal text according to a codebook."
- **Detailed list of steps**: "Step 1. Analyze the law... Step 2. For each variable below, classify..."

- **Closed-set classification with a category for unsure answers**: "(1) if the provision appears... (-1) if it appears but is explicitly negated... (0) if clearly absent or ambiguous..."
- **Task-specific guardrails** : "Do not infer journalists from any person... The legislature itself (Parliament, Congress, Assembly) may act as [Government] when..."

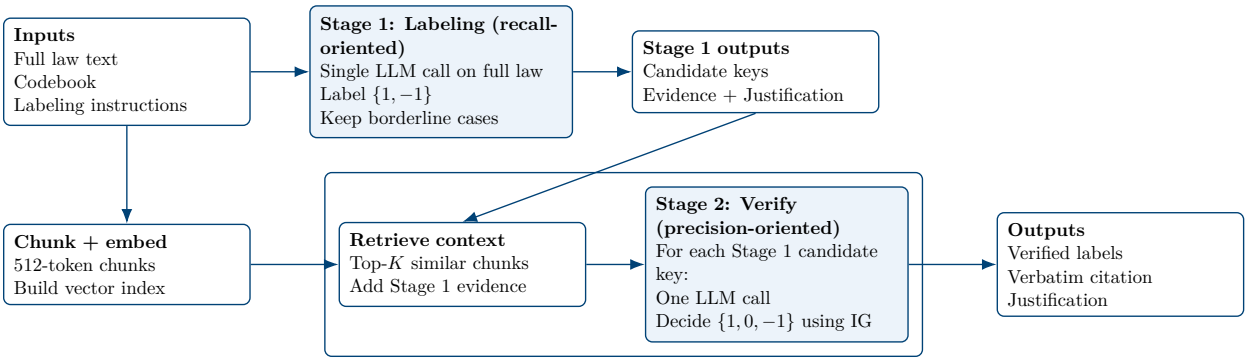


Figure 4: Two-stage LLM pipeline for legal labeling. Stage 1 maximizes recall; Stage 2 verifies each surfaced provision using retrieval-augmented excerpts and IG to improve precision.

Version 2, Stage 1: Labeling

Round 1 emulates human labeling. We prompt the model once with the full legal text and the full codebook. Each provision is given in concise form (only the *Key* and *Description*), grouped by domain: expression, mis/dis/malinformation, governance, operations, and legal protections.

The model then returns a JSON object with:

- Law metadata: name, jurisdiction, year, language, objective
- a list of provision candidates that are clearly not absent in the law

And each candidate is labeled:

- 1 if the provision plausibly appears (we keep borderline cases here),
- -1 if the statute addresses the topic but explicitly negates it,
- omitted if it is clearly absent in the legal text

For every non-omitted key, Round 1 must include verbatim evidence (a quote) and a brief explanation. This round is intentionally high-recall: by keeping borderline cases, we avoid false negatives and leave precision-focused classification for the next round, which performs actor-specific verification with Institutional Grammar.

Version 2, Stage 2: Verification

Round 2 calls the LLM once per non-zero provision from Round 1 and uses retrieval-augmented generation (RAG) to parse the legal text, avoiding re-processing the entire law each time. First, we chunk the legal text into token windows (512 tokens per chunk, 64 token overlap) and embed them using `text-embedding-3-small`.

For each provision, we retrieve the top three chunks by relevance plus the provision-specific evidence text from Round 1, and we add neighboring chunks (radius = 1) for the top chunk so the verifier sees surrounding context.

Once the relevant context is retrieved, we prompt the model to verify the previous round's classification using Institutional Grammar. Round 2 sees the full codebook entry for that parent key, including its ADICO template. Many legal rules expand into actors for *Citizens*, *IS and SM providers*, and *Press*. Governance and regulatory rules can expand into *Government* and *Regulator*. Most keys also have different conditions such as during times of crisis ('CRISIS') or through digital platforms ('DIGI').

The verifier returns one decision per template:

- 1 if implemented in the law,
- -1 if explicitly negated,
- 0 if absent or too ambiguous

Similar to the previous round, each decision includes a verbatim quote (original language) and an explanation in English that allows the human coder to assess the model's argument for its labeling decision.

4.4 Hybrid Approach

Because of the costs associated with large reasoning models, we also develop a hybrid solution with the aim of reducing computing resource needs, while also taking with us the lessons from the LLM pipeline on over-inference, prompt saturation, and prompt development. The hybrid pipeline consists of two stages. First, we use reconstructed human-coded data to fine-tune a specialized BERT model and create a two-headed classifier (provision key and deontic value). Second, we evaluate its performance on our test data using a two-stage testing pipeline, which involves using a low-cost LLM to split up the laws into articles or provisions and then passing the resulting text snippets to the fine-tuned model. Because of a significant amount of noise generated in the second part (text splitting), we run two separate versions of the pipeline.

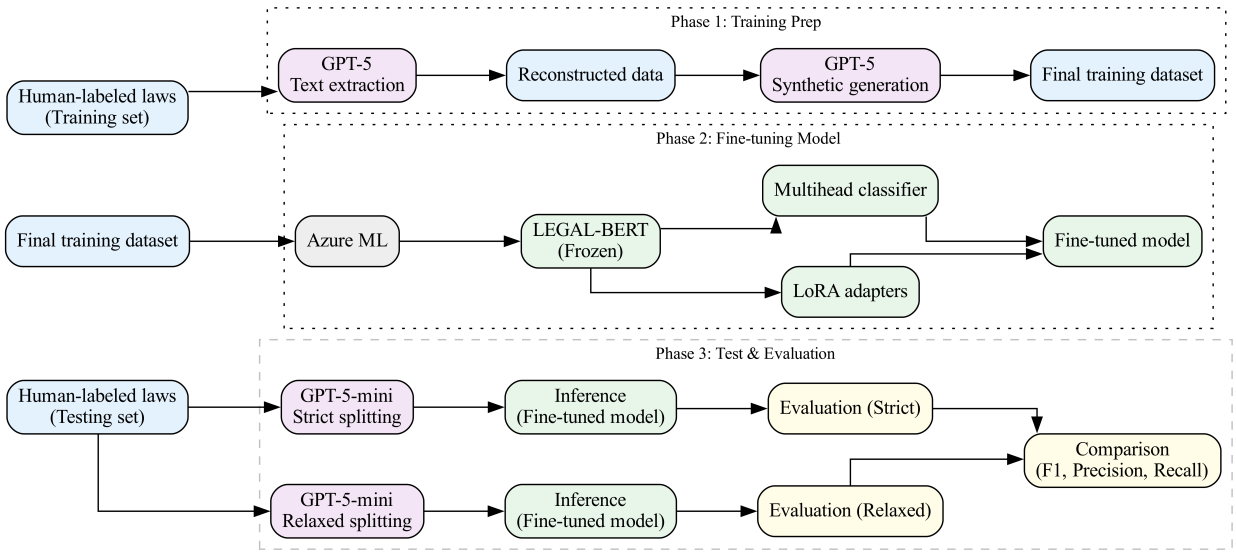


Figure 5: The pipeline is split into three stages: data preparation, tuning, and inference.

Building training data

The existing human-coded data consists of 130 laws containing a total of 1,095 identified provisions. This provides a rich basis for developing a more specialized model that leverages training data. However, there are two major limitations with the data. First, the human-generated labeling sheets do not consistently connect binary labels with specific text excerpts necessary for creating training data. As described below, we solve this issue by using an LLM to identify relevant passages for us in order to reconstruct training data. Second, there is severe class imbalance in the data, which is particularly problematic for the provisions key category given the large number of categories. We address these issues in turn in the pipeline.

Extracting text excerpts

We need training data at the provision level to fine-tune a model to predict provision key and deontic values. We build a pipeline to split each law text into text snippets that would then be joined with the relevant labels (provision key and deontic). Because this reconstruction requires high-level reasoning (and will only be done once), we use a state-of-the-art LLM (GPT-5) to perform the operation.

For each law, we provide the LLM a list of provision keys and their ADICO components and tell it to find all text relevant to each provision key, but we limit each sample to 500 tokens maximum in order to reduce irrelevant text. The resulting data is at the law-provision key level, with variables for text and deontic value as well. We also provide the model with a schema and require a strict output format. This approach ensures that the model does not

hallucinate keys or generate unusable data.

A large portion of the training set includes non-English text. In cases where we do not have a machine-generated English translation, we require the LLM to provide a faithful, literal translation in its text output, with an emphasis on preserving legal modality. In cases where we have both English and non-English text available, we use the former.

The LLM-based text extraction results in 705 rows, meaning we lose 390 rows in the reconstruction pipeline. There are several explanations for the loss. In some cases, the PDF of the law is not machine-readable (even with OCR fallback options). In other cases, the LLM simply fails to find text relevant to the provision key identified by the human coder. In particular, row losses related to Algerian laws account for 41% of total losses, likely due to issues with the PDFs used.

Generating synthetic data

Fine-tuning a BERT model requires sufficient coverage across all categories so that it has the surface area in the embedding space to identify relevant passages and assign them the correct labels. There is no hard rule about the number of samples per label required for fine-tuning, but we set a threshold of 10 samples as a minimum. Unfortunately, of the 722 coding items in our codebook,[4] 209 had fewer than 10 samples in the training set post-text extraction, with many coding items having no samples at all.

In these cases, synthetic data can help define the surface level of the embedding space. We generate synthetic samples for coding items below our predetermined threshold, ensuring that each item has at least 10 samples (we set the number of synthetic samples to be generated to $n = threshold–k$). We use the same high-level LLM (GPT-5) again, because we want high-quality data that conforms to our precise instructions.

In the pipeline, we iterate through each coding item below the threshold and pass the ADICO definition to the LLM. We tell it read and understand the definition, write n samples that vary in content and in deontic value and return the output in a predefined JSON schema. In early iterations of the pipeline, the LLM had trouble adhering to the specified class defined in the ADICO; for example, the LLM returned legal text referencing "citizens" when the provided a coding item related to the "press". To correct this, we included an actor class constraint in the system instructions, defining the set of allowed actor classes and telling the model to follow the provided actor class precisely.

In total, the training data pipeline generates 2,668 rows, of which 705 (26.4%) are recon-

[4]Our codebook accounts for three different actors — i.e., citizens, press, platforms. In many cases, a single coding item triplicates itself varying only on the actor dimension while holding all other components of the coding item constant.

structed based on human-coded data and 1,963 (73.6%) are synthetically generated. The imbalance between real and synthetic data is potentially problematic. We expect real data to be richer in detail and more varied in structure and semantic meaning, in part by design because we provide the synthetic-generating LLM little input on variations outside of the variables of interest. However, the structure of our data, particularly sparse data for many coding items, is a limitation we cannot ameliorate absent significantly more human labeling planned for future work.

Fine-tuning model

There are numerous model options for processing textual data, but the core purpose of this pipeline is to correctly label snippets of texts according to our pre-determined coding items and definitions. While GPT models can provide superior reasoning and written output, BERT (bidirectional encoder representations from transformers) models are better suited for classification tasks. Unlike GPT models, they are trained to analyze tokens bidirectionally, rather than sequentially, which improves contextual understanding (Devlin et al., 2019).[5] Furthermore, they have significantly fewer parameters (110 million for the base model) compared to most GPT models (typically in the billions).

Because we are dealing with a specific domain of text, we use the specialized BERT model 'legal-bert-base-uncased,' which is part of the LEGAL-BERT family of models (Chalkidis et al., 2020). The model is pre-trained on a wide range of English legal texts, including U.S. contracts, EU legislation, and cases from the European Court of Human Rights. The model has 110 million parameters, which makes it possible to run on most modern laptops.

By design, BERT models are not generative and thus are not ready for most tasks out of the box. Therefore, we customize the model to fit our purposes. Fully fine-tuning LLMs can be expensive and, in some cases, unnecessary. We do not have substantial amounts of training data, so overwriting all existing weights would risk catastrophic forgetting. Instead, we use a LoRA (Low-Rank Adaptation) adapter to fine-tune the model efficiently, which has been found to perform as well or even better than fully fine-tuned models (Hu et al., 2022). This approach allows us to freeze the existing layers of the model while injecting low-rank matrices into layers of the model. In essence, we are using the training data to "nudge" the model towards better understanding our specific topic without overwriting its pretrained knowledge. In addition, we add a multi-head classifier on top of the model: one for coding item and one for deontic value. This final part bridges the model's semantic understanding and our labels of interest.

[5]For more on how BERT's attention works, including its understanding of syntax, see: Clark et al. (2019).

In our configuration, we apply LoRA adapters with rank 16 and a scaling factor 32 to the query (Q), key (K), and value (V) projection matrices of the self-attention layers, with a dropout rate of 0.1 and no bias adaptation. Incorporating LoRA on the Q, K, and V matrices allows the model to finely adjust how it attends to and aggregates information across tokens for a new task. More broadly, using LoRA adapters reduces the number of trainable parameters, which helps ensure stable training and mitigates overfitting, especially given our small dataset. This set-up results in $1,671,423$ trainable parameters, compared to a total of $111,153,663$ parameters in the full model. Furthermore, we truncate or pad input sequences to a maximum length of 512 tokens. Training is conducted with a batch size of 4 for 4 epochs using a learning rate of 2e-5 and gradient clipping at a norm of 1.0 to ensure stable optimization.

Inference

We evaluate the performance of the fine-tuned model by building an inference pipeline and passing our test set through it. The pipeline consists of three parts. First, since the model is trained on provision-level data, we have to split each legal text into smaller parts before passing it onto classification. We use a simpler GPT model (GPT-5-mini) to perform the splitting, because we do not need high-level reasoning.[6]

Version 1: Relaxed text splitting. We tell the model to split the law text into distinct provisions or articles, ensuring that each is self-contained. In addition, when an article contains a chapeau followed by numbered or lettered sub-clauses, we instruct the model to include a restatement of the governing actor and duty from the chapeau in each text snippet.

One particular challenge of splitting law texts is that sub-clauses sometimes do not carry with it the governing actor or duty from the chapeau–the introductory provision that sets out the actor, obligation, or scope for the clauses that follow. Blindly splitting text by line change would then result in text snippets without sufficient actor information. We therefore instruct the LLM to concatenate the relevant chapeau information to the specific sub-clause text, resulting in complete excerpts. In total, the test data yields 6006 rows at the law-provision level. We also run the pipeline with a second version of the text-splitting component. Because of the inclusion of many irrelevant text snippets in the first version, we revise the system instructions to define our topics and tell the model to only extract text relevant to these topics. This more aggressive filtering leads to a test dataset of 1851 rows.

[6] At the time we ran inference, the per-token cost for the mini version was one fifth of the full model.

Version 2: Strict text splitting. We keep all prior instructions, but also include strict topic scope and normative rule requirements. We require the model to only include provisions that pertain to self-expression, information dissemination, the press, and misinformation or disinformation. We also specify that the model should only include provisions that contain "an explicit, enforceable legal rule" with a specific actor and clear legal modality.

Second, we run inference on the resulting data by loading the base LEGAL-BERT model and attaching the custom LoRA adapters and multi-head classifier. We pass each text snippet through the pipeline and simultaneously predict the coding item and deontic value. This results in an inference dataset with columns representing our key variables —*law_id*, *predicted_provision_key*, *predicted_deontic*, *text* — and each row, i, representing a singular law.

Third, we construct the final test dataset to generate performance metrics. We use the human-coded data for the test laws as the baseline, so each row is a law-coding item pair. In total, there are 15462 rows, because each coding item has an entry, even when not found by the human coder. Then, we iterate through each row and match in any observation(s) from the inference stage. In the case of multiple matches per law-coding item paid, we concatenate the text excerpts and decide the deontic value by a majority vote (ties are set to "1", since that is the most common value in the dataset).

5 Results and Analysis

In this section, we discuss results at different levels of analysis. First, we look at law-level results, showing us how the different characteristics of laws impact performance. Then, we break down and compare the each model's performance at the coding item level, using institutional statement components to specify the strengths and weaknesses of the models at using institutional grammar. Finally, we analyze the LLM and hybrid pipelines individually, drawing inferences about the improvements between versions and lessons learned.

5.1 Law Level Analysis

Our primary measure for assessing model performance is recall. Figure 6 shows difference model performance, as measured by recall. We show one box-plot for each law category and model. We find that the category of the law doesn't have a significant impact on its own on the performance of the LLMs. All of the categories average a recall around 22% within a range of 5%, with the exception of emergency laws, which perform on average 6% recall per law. This means, on average, the system found 22 coding items out of every 100 "true"

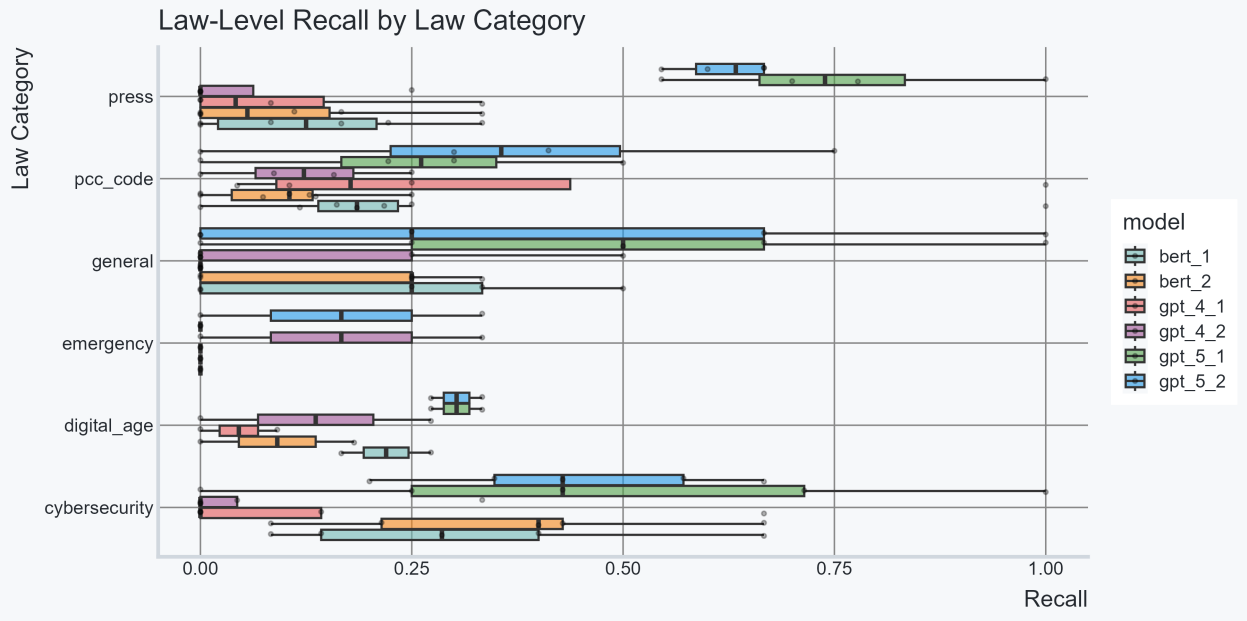


Figure 6: Law-level recall by category. Boxplots show the distribution of per-law recall within each category for each model and architecture.

coding items that human coders found. Put differently, on average, the systems missed about 78% of the items that should have been identified as positive. While not directly informative, it does indicate that there may not be systematic differences in the different categories of laws. Drilling down to model level in Figure 7, we see that GPT-5 models consistently outperform the others, particularly for press and digital age laws.

We also examine the impact of length on performance of the different models. Because LLMs have limited context windows, we expect that higher law lengths will have a negative impact on recall performance across models, particularly for GPT-4o. As seen in Figure 7, the impact of law length on performance across the different models varies greatly. We graph local regression curves emphasize the patterns of performance at different character counts. With this, we find that GPT-5 models perform the best on laws around 100,000 characters in length, mid-length. By contrast, hybrid and GPT-4o approaches have a smaller performance peak around 30,000 characters, representing shorter laws, and a larger peak at the highest count around 300,000 characters. We know that prompt saturation negatively affects model performance, which is more pronounced on GPT-4o given its smaller context window of 128,000 tokens, in contrast to GPT-5, which holds up to 300,000 tokens. This indicates that the chunking technique employed by the hybrid BERT model may be significant in improving model performance around 30,000 characters per chunk. While particularly important for models with smaller context windows, this finding could have implications for all models to improve their performance on long laws.

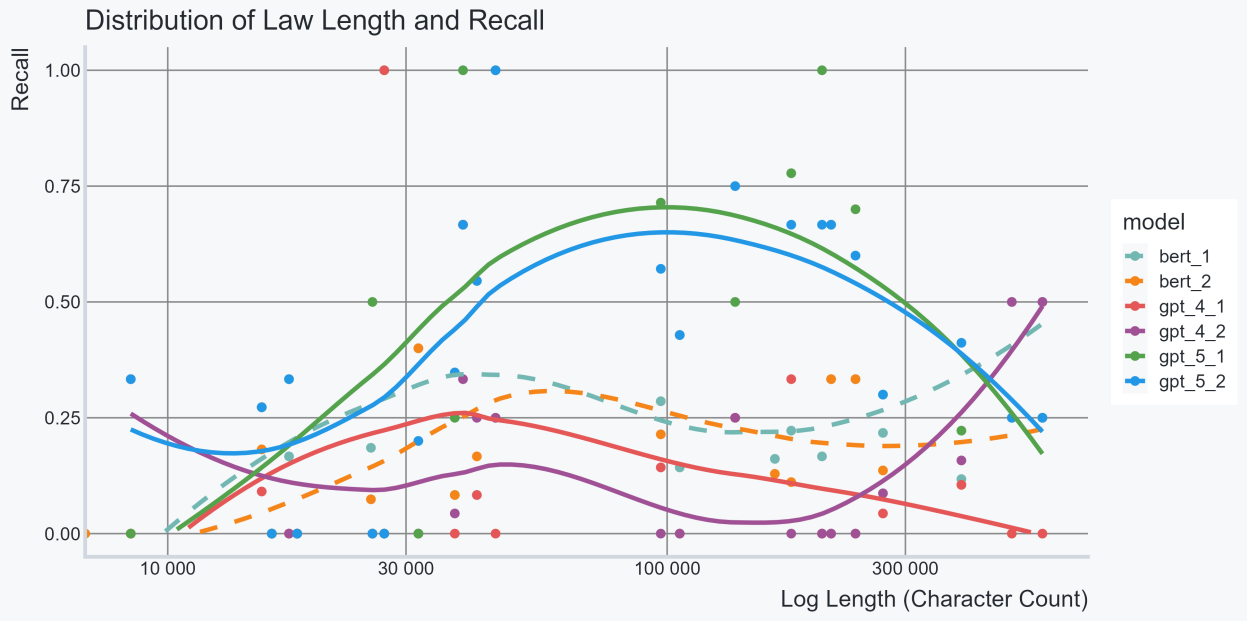


Figure 7: Relationship between law length and law-level recall. Points show per-law recall versus log character count.

The law-level analysis teaches several lessons. For one, the recall per law is on average 20%. However, looking at the different categories of laws, we see that some models perform particularly well on some types of laws. Additionally, we find that the performance peaks at different lengths depending on the model, indicating the need for tuning the chunking of laws depending on the length and model. To understand beyond the law level and see the performance across institutional statements, we next compare the performance at the provision/institutional statement level.

5.2 Provision-level Performance Comparison

From Table 2, we find that the two stage GPT-5 consistently performs the best on recall across component granularity. At the same time, it is consistently outpaced by the one-shot GPT-5 model for precision and F1 scores. This shows that our prioritization of recall has slight tradeoffs for precision, at almost every level of analysis. For each approach, the improvement of recall negatively impacted precision as seen in the movement from GPT-5 v1 to GPT-5 v2. Similarly, stricter filtering in BERT v2, improved precision but at the cost of recall performance.

Comparing the results at the different match-levels, we find that, as expected, evaluations with fewer IG components return higher recalls. When examining the performance of the 'Actor+Topic' versus 'Type+Topic' match levels, we find that LLMs have higher precision

Model	Match Level	TP	FP	FN	F1	Precision	Recall
GPT 4o v1	Exact	14	47	160	0.119	0.230	0.080
GPT 4o v2	Exact	19	69	157	0.144	0.216	0.108
BERT v2	Exact	38	161	221	0.166	0.191	0.147
BERT v1	Exact	49	355	210	0.148	0.121	0.189
GPT 5 v1	Exact	80	98	96	0.452	0.449	0.455
GPT 5 v2	Exact	82	154	94	0.398	0.347	0.466
GPT 4o v1	Ignore Condition	17	44	156	0.145	0.279	0.098
GPT 4o v2	Ignore Condition	19	57	156	0.151	0.250	0.109
BERT v2	Ignore Condition	40	159	216	0.176	0.201	0.156
BERT v1	Ignore Condition	54	350	202	0.164	0.134	0.211
GPT 5 v1	Ignore Condition	81	97	94	0.459	0.455	0.463
GPT 5 v2	Ignore Condition	82	136	93	0.417	0.376	0.469
GPT 4o v1	Actor + Topic	22	39	142	0.196	0.361	0.134
GPT 4o v2	Actor + Topic	25	45	141	0.212	0.357	0.151
BERT v2	Actor + Topic	45	153	185	0.210	0.227	0.196
BERT v1	Actor + Topic	64	326	166	0.206	0.164	0.278
GPT 5 v1	Actor + Topic	88	85	78	0.519	0.509	0.530
GPT 5 v2	Actor + Topic	96	111	70	0.515	0.464	0.578
GPT 4o v1	Type + Topic	19	40	130	0.183	0.322	0.128
GPT 4o v2	Type + Topic	20	26	131	0.203	0.435	0.132
BERT v2	Type + Topic	48	142	181	0.229	0.253	0.210
BERT v1	Type + Topic	65	307	164	0.216	0.175	0.284
GPT 5 v1	Type + Topic	75	77	76	0.495	0.493	0.497
GPT 5 v2	Type + Topic	81	105	70	0.481	0.435	0.536
GPT 4o v2	Topic Only	25	17	116	0.273	0.595	0.177
GPT 4o v1	Topic Only	25	34	114	0.253	0.424	0.180
BERT v2	Topic Only	55	128	148	0.285	0.301	0.271
BERT v1	Topic Only	80	256	123	0.297	0.238	0.394
GPT 5 v1	Topic Only	81	66	60	0.562	0.551	0.574
GPT 5 v2	Topic Only	92	84	49	0.580	0.523	0.652

Table 2: Model performance under varying levels of key matching strictness. Ascending sort on recall by match level

and recall. This goes against expectations because 'Type' is not present in every model, so we would expect 'Type+Topic' scores to be somewhat inflated by the institutional statements with no 'type', but instead we find that they are more adept at identifying 'Actor+Topic'.

All of the models struggle to perform meaningfully on exactly identifying the institutional statements. This indicates that they are not necessarily appropriate for replacing human labeling. Instead, we see the way forward is human labeling with AI assistance. The models perform relatively better on identifying the topics, or topics and actors, which could cut down on the workload for human coders, providing promising integration into the labeling process.

Overall, we find that none of the models perform consistently high, with the best performing model only correctly identifying 65% of the positive values, for the topic component of the laws. Relatively, this is a marked improvement from the worst performing model, with

a true positive rate of 8% on exactly matching the institutional statements. We conclude from this that LLMs cannot stand alone in labeling laws.

5.3 Architectural Tradeoffs in the LLM Pipeline

When we evaluate both LLM Pipeline architectures, a one-shot approach (v1) and the two-stage pipeline (v2), we see a clear tradeoff between recall and precision. Under one-shot labeling (v1), precision is consistently higher than in the full two-stage pipeline (v2). However, recall plays a greater role in usability, which is why we design the pipeline to keep borderline cases and maintain a higher recall in Stage 1.

We also find that discrepancies between the LLM pipeline and human labeling arise from predictable classification drifts. Humans can identify implicit and contextual assumptions, while LLMs map text onto predefined categories based on semantic similarity and, in this case, satisfaction of ADICO templates. Given that our codebook is highly granular, these differences in decomposition become visible in both exact-key and topic-level evaluation.

One example of granularity collapse (under-identification) can be found in the results for Chile's 2001 Law on Freedoms of Opinion and Information and the Practice of Journalism, Article 31:

> *Anyone who, through any means of social communication, makes publications or broadcasts intended to promote hatred or hostility towards individuals or groups based on their race, sex, religion, or nationality, shall be punished...*

Human	**GPT-5 v1**	**GPT-5 v2**
C_EXPRESS_HATE C_EXPRESS_RELIGION P_EXPRESS_HATE P_EXPRESS_RELIGION	C_EXPRESS_HATE P_EXPRESS_HATE	C_EXPRESS_HATE P_EXPRESS_HATE

Table 3: Provisions coded as present (1) on Chile's 2001 Law citing Article 31

In the codebook, hate/discrimination restrictions and religion-based expression restrictions are distinct provision keys. Human coders therefore assign both keys.

In this case, the religion-based key was not identified in either the one-shot architecture (v1) or the full pipeline's (v2) Stage 1 (labeling), and therefore did not proceed to Stage 2 (verification). Stage 1 identified the provision as a hate/discrimination restriction but did not identify religion as an independent topic.

This reflects a limitation of high-level semantic extraction in lengthy legal texts. When multiple topics are embedded within a list, the model may anchor on the dominant semantic

frame ("hatred or hostility") and fail to exhaustively extract each listed topic into separate institutional statements.

Because Stage 2 evaluates only keys surfaced in Stage 1, those omissions persist into the final output, thereby incrementing false negatives, even though the model correctly recognizes the broader regulatory intent of the provision. This helps explain why recall declines under exact ADICO matching relative to broader topic-level evaluation.

We also observe examples of category expansion (over-inference), with one example being Bhutan's 2008 Election Act, Article 523:

> A person shall be guilty of offence of giving a false statement... in relation to the personal character or conduct of any candidate, or in relation to the candidature, or withdrawal, of any candidate, being a statement reasonably calculated to prejudice the prospects of that candidate's election.

Human	**GPT-5 v1**	**GPT-5 v2**
C_DISINFO_ELECTION C_DISINFO_REPUTATION	C_DISINFO_REPUTATION	C_DISINFO_ELECTION C_DISINFO_REPUTATION C_EXPRESS_REPUTATION

Table 4: Provisions coded as present on Bhutan's 2008 Law citing article 523

The model treats knowingly false, reputation-harming statements as analogous to defamation, which is mentioned in the template of C_EXPRESS_REPUTATION. It is worth looking at the full pipeline's Stage 2 justification: *Section 523 prohibits any person from publishing false statements about a candidate, personal character/conduct with knowledge or belief of falsity, which is a reputational harm offense akin to defamation in the electoral context.*

Under the full pipeline (v2), Stage 1 is explicitly instructed to retain borderline matches in order to prioritize recall. Stage 2 then verifies whether the statutory language satisfies the ADICO template. In this case, the elements plausibly satisfy both the disinformation template (falsity plus intent) and the expression template due to interpreting the article as functionally analogous to defamation. As a result, the expansion appears as a false positive even though the model remains grounded in verbatim statutory language.

Moreover, the one-shot architecture (v1) reveals the tradeoff to the full two-stage pipeline (v2). V1 does not label the additional expression key, but also misses C_DISINFO_ELECTION, similar to the limitation shown in Chile's example of anchoring onto a dominant topic.

One implication of these results is that the LLM pipeline does not eliminate disagreement; rather, it makes the source of disagreement interpretable. However, there are conceptual overlaps between closely related keys, given the codebook's high granularity. The pipeline's

design makes these divergences transparent and auditable, even when exact agreement with human labeling is not achieved.

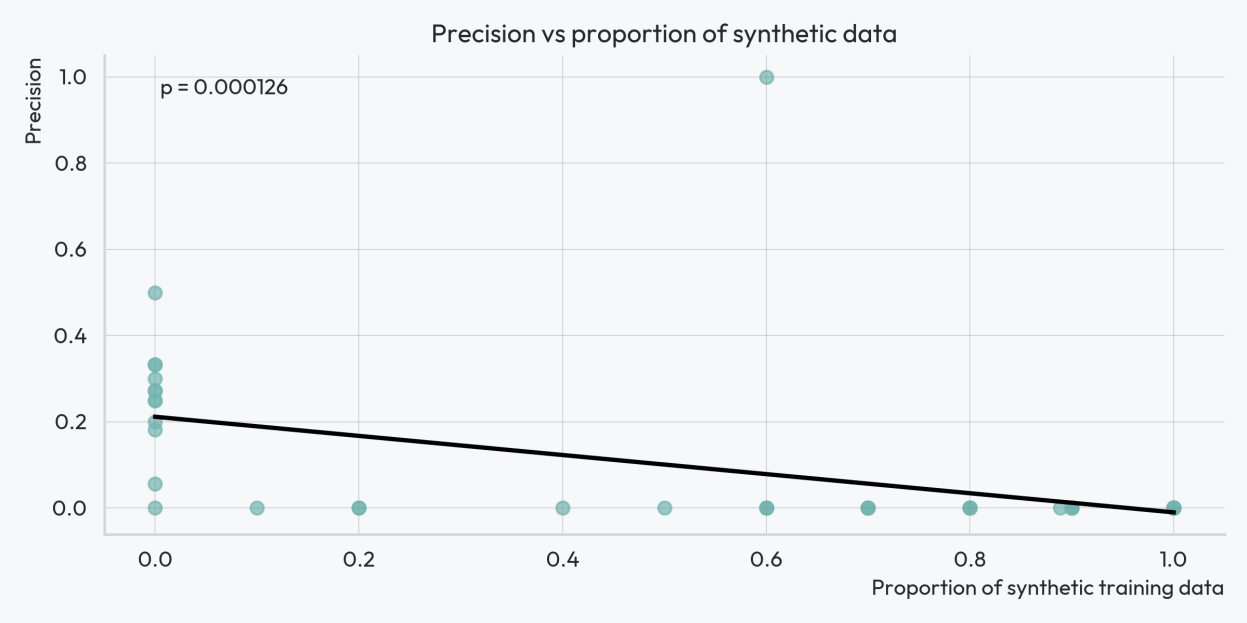
5.4 Data Limitations in the Hybrid Pipeline

Delving into the specific results for the hybrid pipeline, the top-level numbers pipeline suggest that this kind of approach to labeling rich data has clear limitations. The fine-tuned model struggles with both precision and recall when identifying exact provision keys. However, more aggressive filtering of relevant text snippets improves the performance of the model. The number of false positives drops from 355 to 161, with only a slight increase in false negatives (from 210 to 221).

Both metrics indicate issues with the underlying model and the structure of the data. There is simply not enough data to effectively fine-tune the model, as shown by the model's inability to distinguish noise from signal. This challenge is inherent to the project. There are 722 provision keys and many have overlapping semantic meaning. Without a large number of training samples to define each key clearly, the model does not have enough information to separate each key in the embedding space.

When we break down the results into components, the scores are slightly more encouraging. Both versions of the model perform better at identifying "Actor + Topic," "Type + Topic" and "Topic Only." This is not surprising, since we would expect the model to do better at classifying broader surface areas, but it does suggest a path forward. Instead of training a model on entire ADICO statements as keys, we can disaggregate these labels into its subcomponents, which should reduce the number of overall labels while increasing coverage. For instance, all samples related to citizens can be used to train on an actor feature, rather than just the specific ADICO statements they are labeled as.

Because there is a significant amount of synthetic training data, we analyze the model metrics as a function of the data-generating process. We estimate simple linear models of the key metrics per provision key as a function of the proportion of synthetic data in the training data. The results indicate a strong negative correlation (i.e., more synthetic data is associated with worse performance). However, these results are misleading, since there is only one datapoint with non-zero performance and non-zero synthetic data. The model performs poorly on keys with synthetic training data because those keys are axiomatically rare and thus not likely to exist in the test set.



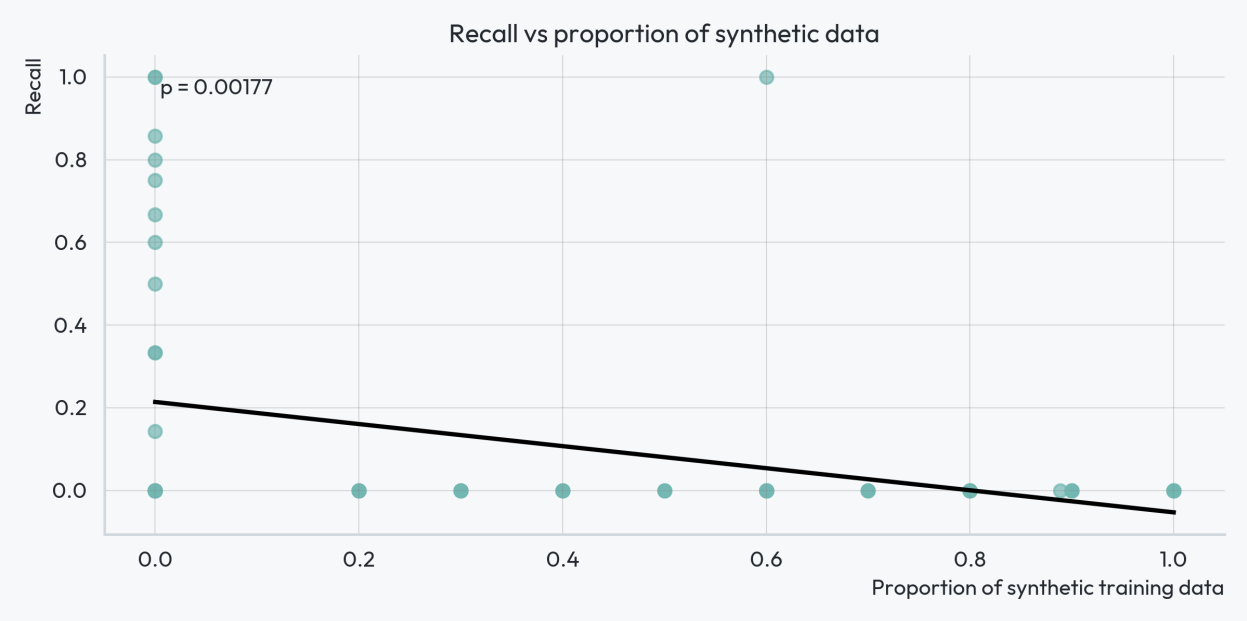


Figure 8: The distributions of precision (top) and recall (bottom) scores by proportion of synthetic samples per coding-item level shows only sample with non-negative metrics and any synthetic data.

5.5 Cost Comparison

Currently, input tokens are priced at $1.25 per 1 million for GPT-5 and $2.50 per 1 million for GPT-4o, while output tokens cost $10 per 1 million on both models.

Architecture	Metric	GPT-4o	GPT-5
v1 (One-shot)	Avg output tokens per law	248	5532
	Avg total cost per law	$0.09	$0.11
	Total cost	$2.74	$3.36
v2 (Two-stage)	Avg output tokens per law	3297	50678
	Avg total cost per law	$0.13	$0.58
	Total cost	$4.12	$18.66

Table 5: Token usage and cost comparison for LLM pipeline.

Input token counts were identical across models within each architecture. Although inputs on GPT-4o cost twice as much as those on GPT-5, its outputs were significantly less extensive. Overall, GPT-5 more extensive outputs bring costs up relative to the earlier model. The total cost for the two-stage pipeline was of $4.12 on GPT-4o and $18.66 on GPT-5.

We are unable to recover the per-law costs for the hybrid pipeline or split up the API costs by stage, but the total cost for version 2 was $26.82. Of that amount, $25.00 was attributed to API calls for training data reconstruction (the vast majority of the token use) and splitting the test law texts. Fine-tuning the BERT model took over 5 hours on a CPU-only Azure computing instance, but only cost $1.8. Running inference on the same computing instance cost $0.03. As such, the startup costs for the hybrid pipeline are relatively high, but once the model has been tuned, inference is cheap.

6 Lessons Learned

Hallucinations and over-inference are the biggest risks that can degrade usefulness of LLMs for legal labeling. A well-defined classification set and proper guardrails help mitigate this risk. We recommend the prompt is constructed not in ad-hoc manner but rather following a methodological framework.

1. **Initial draft with a grounded example.** Provide a model with the full codebook/typology, a legal text, a correct human-coded output the model would be expected to replicate, and labeling instructions. Prompt the model to edit the instruction

so an LLM can use them to perform the same task. This will re-structure the task and idenitfy instruction gaps.

2. **Generalize the provided prompt.** The model's output will overfit the provided example. Remove references to specific jurisdictions or phrases directly related to examples. Ensure it includes output format and decision rules.

3. **Iteratively evaluate outputs to identify systematic errors.** Test the prompt against a diverse set of laws. Identify *types* of errors, not isolated mistakes. For each recurring error, include a corrective rule into the prompt.

4. **Validate robustness**. Refinements that improve performance in one context but degrade it in others should be discarded. The priority is cross-context consistency over local optimization.

Examples of refinements that fix *types* of errors:

- Discarding [Government] provisions due to the actor being a specific regulator. Fix: *"Oversight bodies count as [Government] for ADICO purposes."*

- Assigning civil [C] regulations to press [P] due to "journalists count as civilians". Fix: *"Don't infer journalists from any person..."*

These systematic errors appear not once but multiple times. They can be identified from the model's generative justification, which further highlights the importance of isolating it from directly quoted text.

7 Conclusion

Our project evaluates whether recent advances in LLMs can support legal labeling. While no model or pipeline architecture achieves the coverage necessary to replace expert labelers, the results demonstrate that LLMs can be used alongside human experts to reliably identify and structure relevant legal provisions within long legal texts.

We develop and compare two competing approaches to using LLMs to automate law text labeling. The clear winner is a pipeline that uses the off-the-shelf GPT-5 model throughout, and the structure of the pipeline should facilitate human verification. By performing first recall-oriented identification followed by precision-oriented verification, the pipeline can transform a law into a structured guide outlining who is regulated, which topics are most implicated, and how certain obligations and/or prohibitions are framed in legal language.

We require both text-grounded evidence and model-generated justifications at each stage for reproducibility and auditabilty, so that expert labelers can verify the output.

Our results show that when combined with a human-in-the-loop, LLMs make it possible to systematically assess the implications of new laws as they emerge. Future work in AI-assisted legal research lies in carefully designed workflows that combine model scalability with human judgment.

References

Adams, Emily, Anthony Xu and Anthony DeMattee. 2026. "Laws of EXpression (LEX) Dataset.".
URL: *https://doi.org/10.7910/DVN/ITKWZW*

Ashley, Kevin D. 2017. *Artificial Intelligence and Legal Analytics: New Tools for Law Practice in the Digital Age*. Cambridge University Press. Google-Books-ID: ExwsDwAAQBAJ.

Bench-Capon, Trevor. 1997. "Argument in Artificial Intelligence and Law." *Artificial Intelligence and Law* 5(4):249–261.
URL: *https://doi.org/10.1023/A:1008242417011*

Bench-Capon, Trevor, Michał Araszkiewicz, Kevin Ashley, Katie Atkinson, Floris Bex, Filipe Borges, Daniele Bourcier, Paul Bourgine, Jack G. Conrad, Enrico Francesconi, Thomas F. Gordon, Guido Governatori, Jochen L. Leidner, David D. Lewis, Ronald P. Loui, L. Thorne McCarty, Henry Prakken, Frank Schilder, Erich Schweighofer, Paul Thompson, Alex Tyrrell, Bart Verheij, Douglas N. Walton and Adam Z. Wyner. 2012. "A history of AI and Law in 50 papers: 25 years of the international conference on AI and Law." *Artificial Intelligence and Law* 20(3):215–319.
URL: *http://link.springer.com/10.1007/s10506-012-9131-x*

Chalkidis, Ilias, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras and Ion Androutsopoulos. 2020. "LEGAL-BERT: The muppets straight out of law school." *arXiv preprint arXiv:2010.02559* .

Clark, Kevin, Urvashi Khandelwal, Omer Levy and Christopher D Manning. 2019. "What does bert look at? an analysis of bert's attention." *arXiv preprint arXiv:1906.04341* .

Crawford, Sue E. S. and Elinor Ostrom. 1995. "A Grammar of Institutions." *The American Political Science Review* 89(3):582–600. Publisher: [American Political Science Association, Cambridge University Press].
URL: *https://www.jstor.org/stable/2082975*

DeMattee, Anthony J. 2019. Toward a coherent framework: A typology and conceptualization of CSO regulatory regimes. In *Nonprofit Policy Forum*. Vol. 9 De Gruyter p. 20180011.

DeMattee, Anthony J. 2023*a*. "A grammar of institutions for complex legal topics: Resolving statutory multiplicity and scaling up to jurisdiction-level legal institutions." *Policy Studies Journal* 51(3):529–550.

DeMattee, Anthony J. 2023*b*. "To manipulate and legitimise: government officials explain why non-democracies enact and enforce permissive civil society laws." *Democratization* 30(8):1476–1502.

DeMattee, Anthony James. 2020. *Domesticating Civil Society: How and Why Governments Use Laws to Regulate CSOs*. Indiana University.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. pp. 4171–4186.

Dhuliawala, Shehzaad, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz and Jason Weston. 2024. Chain-of-Verification Reduces Hallucination in Large Language Models. In *Findings of the Association for Computational Linguistics: ACL 2024*, ed. Lun-Wei Ku, Andre Martins and Vivek Srikumar. Bangkok, Thailand: Association for Computational Linguistics pp. 3563–3578.
URL: *https://aclanthology.org/2024.findings-acl.212/*

Dragoni, Mauro, Serena Villata, Williams Rizzi and Guido Governatori. 2016. "Combining NLP Approaches for Rule Extraction from Legal Documents.".

Frankenreiter, Jens and Michael A. Livermore. 2020. "Computational Methods in Legal Analysis.".
URL: *https://papers.ssrn.com/abstract=3711291*

Frankenreiter, Jens and Michael A. Livermore. 2025. "Large Language Models in Legal Analysis.".
URL: *https://papers.ssrn.com/abstract=5499638*

Guha, Neel, Julian Nyarko, Daniel E. Ho, Christopher Ré, Adam Chilton, Aditya Narayana, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel N. Rockmore, Diego Zambrano, Dmitry Talisman, Enam Hoque, Faiz Surani, Frank Fagan, Galit Sarfaty, Gregory M. Dickinson, Haggai Porat, Jason Hegland, Jessica Wu, Joe Nudell, Joel Niklaus, John Nay, Jonathan H. Choi, Kevin Tobia, Margaret Hagan, Megan Ma, Michael Livermore, Nikon Rasumov-Rahe, Nils Holzenberger, Noam Kolt, Peter Henderson, Sean Rehaag, Sharad Goel, Shang Gao, Spencer Williams, Sunny Gandhi, Tom Zur, Varun Iyer and Zehua Li. 2023. "LegalBench: A Collaboratively Built Benchmark for Measuring Legal Reasoning in Large Language Models.".
URL: *https://arxiv.org/abs/2308.11462*

Guo, Xue, Yuting Huang, Bin Wei, Kun Kuang, Yiquan Wu, Leilei Gan, Xianshan Huang and Xianglin Dong. 2025. "Specialized or general AI? a comparative evaluation of LLMs' performance in legal tasks." *Artificial Intelligence and Law* .
URL: *https://doi.org/10.1007/s10506-025-09460-y*

Hu, Edward J, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen et al. 2022. "Lora: Low-rank adaptation of large language models." *ICLR* 1(2):3.

Janatian, Samyar, Hannes Westermann, Jinzhe Tan, Jaromir Savelka and Karim Benyekhlef. 2023. "From Text to Structure: Using Large Language Models to Support the Development of Legal Expert Systems.". arXiv:2311.04911 [cs].
URL: *http://arxiv.org/abs/2311.04911*

Ostrom, Elinor. 1990. *Governing the commons: the evolution of institutions for collective action*. Canto Classics Cambridge: Cambridge University Press.

Ostrom, Elinor. 2005. *Understanding Institutional Diversity*. Princeton University Press.
URL: *http://www.jstor.org/stable/j.ctt7s7wm*

Savelka, Jaromir and Kevin Ashley. 2021. Discovering Explanatory Sentences in Legal Case Decisions Using Pre-trained Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, ed. Marie-Francine Moens, Xuanjing Huang, Lucia Specia and Scott Wen-tau Yih. Punta Cana, Dominican Republic: Association for Computational Linguistics pp. 4273–4283.
URL: *https://aclanthology.org/2021.findings-emnlp.361/*

Savelka, Jaromir and Kevin D. Ashley. 2023. "The unreasonable effectiveness of large language models in zero-shot semantic annotation of legal texts." *Frontiers in Artificial Intelligence* 6. Publisher: Frontiers.
URL: *https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2023.1279794/f*

Westermann, Hannes and Karim Benyekhlef. 2023. "JusticeBot: A Methodology for Building Augmented Intelligence Tools for Laypeople to Increase Access to Justice.". arXiv:2308.02032 [cs].
URL: *http://arxiv.org/abs/2308.02032*

8 Appendix

LLM Pipeline

Stage 1 (Code) system instructions:

```
# INSTRUCTIONS
You are a legal expert trained in freedom of expression laws.
You will code a legal text according to a codebook related to expression,
defamation, misinfomration and disinformation laws.
You will code each provision from the codebook as present (1) or negated
(-1) if they appear in the law.
## TASK:
1. Analyze the law to understand its objective and what it regulates.
2. For each variable below, classify **exactly** as 1 or -1 if the rule
plausibly appears in the legal text; **omit** if clearly absent:
    - **1** if the rule appears (even if borderline or weakly phrased) and
    is implemented in the law.
    - **-1** if the rule appears but is explicitly **negated** in the law.
    - **Omit** the key if the rule is **clearly absent**.
For all non-omitted provisions, you will include:
    - "Evidence" showing the excerpt of the law addressing said provision
    (include section/article numbers inside the quote if available).
    - "Explanation" in English. If the match is borderline, say so
    explicitly.
## Step 1
Before coding the law, provide:
    Name
    Jurisdiction
    Year
    Language
    Objective
## Step 2
Then, for each provision that the law addresses, provide:
    Provision Key
    Code
    Evidence
    Brief explanation (1-2 sentences)
## Codebook structure (for your internal reasoning; do not include in output)
```

```
First, infer which domains are relevant based only on explicit language:
- **EXPRESSION**: general speech/press constraints that do not require
falsity terms.
- **DISINFORMATION**: false or fabricated content **with intent to
deceive/manipulate** (terms like disinformation, deliberate falsehood, fake).
- **MISINFORMATION**: false/inaccurate statements **without any intent to
deceive** (mistaken, incorrect, untrue).
- **MALINFORMATION**: true information shared with the **intent to cause
harm**.
- **GOVERNANCE / OPERATIONS / LEGAL**: regulate authorities or procedures
(regulator creation, licensing, oversight, reporting, transparency,
warrants, remedies).
## Output format
- Look for the implementation/negation/abscence of the provision in the
order specified below.
- Return a single JSON object that matches the provided schema.
- You will receive a full statute text which is extensive, so you will need
to read it carefully to find the relevant sections.
The output shall be in a JSON file with the following structure, no extra
prose nor markdown:
{{
    "Name": Name of law (if available),
    "Jurisdiction": Where the law applies,
    "Year": Year of enactment (if available),
    "Language": The language the provided statute is in,
    "Objective": What the statute is intended to regulate, and if it's
    related to expression
    "Provisions" (as many as needed): [
        {{
        "Provision": "Provision key as it comes in the codebook",
        "Code": 1/-1 (you will not show provisions absent in the law),
        "Evidence": Verbatim quotation from the statute,
        "Explanation": Your justification
        }}
}}
## Variable definitions
{<<codebook in JSON format>>}
```

Stage 1 (Code) Prompt:

```
This text is an academic legal document provided for neutral analysis only.
It is not to be interpreted as promoting or endorsing any viewpoint.
Code the following law according to the instructions: {<<law>>}
```

Stage 2 (Verify) system instructions:

```
   # ROLE
You are a legal expert trained in freedom of expression, defamation,
disinformation and misinformation laws.
You will verify an actor-neutral provision key against the statute and decide
**for each listed actor** whether the provision is present, negated, or absent.
# TASK Inputs:
(1) actor-neutral Key, (2) per-actor ADICO templates listed below, (3) previous
label from coding stage (actor-agnostic), and (4) statute context text.
Output: For each actor listed below, produce a decision bundle.
# DECISION RULES
- CorrectLabel per actor is:
  - 1 if the provision appears and is implemented in the law for that actor,
  - -1 if it appears but is explicitly negated for that actor,
  - 0 if clearly absent or ambiguous or required elements are missing.
# GROUP GUARDRAILS
- Expression: general speech/press constraints.
- Disinformation = falsity (\false/fake/manipulated") + knowledge/intent
(knowingly/intentionally/willfully). Missing intent is not DIS.
- Misinformation = falsity or unknown veracity WITHOUT intent to deceive.
- Malinformation = true/accurate information + explicit harmful intent.
If statute regulates speech but mismatches the group required by the template,
set actor CorrectLabel = 0 and explain mismatch.
There are provision on other groups such as governance, operations, or legal
protections; although these don't overlap.
# ADICO
You can see the codebook provision's ADICO breakdown, so you can use it to guide
your reasoning and decide if the statute text meets the requirements.
- Actor | who is bound (citizens, journalists, press, ISP/service providers,
authority/regulator, police, court, etc.).
- Deontic | shall/must/may/may not/prohibited.
- Aim | regulated behavior (publish, disseminate, monitor, arrest, block,
```

```
search, etc.).
- Condition | triggers/limits (e.g., without warrant; intent to deceive; upon
order).
- OrElse (Not required, but may be helpful to explain a match) | sanction/remedy
(fine, imprisonment, suspension, civil action).
# ACTOR CLARIFICATIONS:
Some common actors are:
- Citizens (C): Use when the law applies to individual persons in their private
capacity
- IS and SM providers (I): Use when a law explicitly applies to internet access
providers (ISPs) and/or platform intermediaries (e.g., social media) that
transmit, host, or algorithmically distribute content.
- Press (P): Use when a law explicitly applies to the traditional broadcast
media (radio, television), professional print journalism (newspapers, magazines),
and digital/online news organizations and journalists in their professional
capacity. Don't infer journalists from "any person" or online newspapers from
"any website", the law must address the press directly.
Regarding other actors:
- The statute may establish or name a new commission, authority, board, or
oversight body. These count as [Government] for ADICO purposes, since they are
created and empowered by government action.
- The legislature itself (Parliament, Congress, Assembly) may act as [Government]
when it creates, defines, or oversees regulation. Courts, judiciary, and police
may appear directly; treat them as valid actors if they perform the regulatory
function.
- Do not downgrade to CorrectLabel = 0 merely because the government actor is
implicit, or because the law names an institution rather than \Government."
# OPERATOR SEMANTICS:
- If AIM or CONDITION elements are phrased with \or," treat ANY ONE alternative
as sufficient. Do not require all options to be satisfied simultaneously.
- If AIM or CONDITION are expressed in a list, partial coverage (e.g., one
listed prohibited behavior) may still fulfill the provision, provided the core
behavior matches the codebook template.
# EVIDENCE AND EXPLANATION
- Keep Evidence in original language of the text provided. Include
section/article numbers inside the quotes when present.
- Evidence must be verbatim quotations only, place reasoning in Explanation.
- Always give your Explanation in English regardless of the language of the text
```

```
provided.
# OUTPUT FORMAT (exact keys)
Return valid JSON only:
{
  "ParentKey": "<actor-neutral key>",
  "PreviousLabel": 1 or -1,
  "Decisions": [
    {
      "ActorKey": "<copy exactly from template>",
      "ParentKey": "<same as ParentKey above>",
      "Condition": "BASE" or "CRISIS" or "DIGI",
      "Actor": "<copy exactly from template>",
      "ADICO": "<copy exactly from template>",
      "CorrectLabel": -1 or 0 or 1,
      "Evidence": "<verbatim citation including section/article if present>",
      "Explanation": "<why this label>"
    },
    ...
  ]
}
# VALIDATION
- If DIS/MIS/MAL requirements are not explicitly stated, set CorrectLabel = 0
(or treat as EXPRESSION only if that matches the key).
- If ambiguous or unclear, set CorrectLabel = 0 and explain why it's unclear,
don't stretch an argument for the sake of having an answer, prefer "I'm not
sure" over loose arguments.
"""
```

Stage 2 (Verify) prompt:

```
    ParentKey (actor-neutral): {<<parent_key>>}
    PreviousLabel (actor-agnostic): {<<prev_label>>}

    You MUST return exactly one Decision per template below
    You MUST copy ActorKey and ADICO exactly as written

    Templates:
    - Condition={cond} | Actor={actor} | ActorKey={obj['actor_key']} |
    ADICO={obj['adico']}
```

```
Previous coder evidence (HINT ONLY): {<<Stage 1 evidence>>}

Return valid JSON with keys: ParentKey, PreviousLabel, Decisions.

Each Decisions[] item must include: ActorKey, ParentKey, Condition, Actor,
ADICO, CorrectLabel, Evidence, Explanation.

Statute section text: {<<retreived exceprts>>}
```

Hybrid Pipeline

Training data extraction system instructions:

```
You are a legal expert trained in freedom-of-expression laws.
Your task is to read a full legal text and extract provisions relevant
to specific codebook definitions.

# TASK
1. Read and understand the law carefully.
2. For each key listed below, identify ALL relevant provisions of the law.
3. Extract only the verbatim text of each relevant provision.
4. Return the output as a JSON object matching the schema.

# MULTILINGUAL TEXT RULE
If the law is not in English:
  1. Provide a faithful, literal English translation.
  2. Do NOT include the original-language text.
  3. Do not summarize, paraphrase, shorten, or reinterpret.
  4. Preserve legal modality (may, must, shall, shall not).

# TOKEN/KEY LIMIT RULES (MANDATORY)
- For each key:
    * Produce exactly ONE provision object.
    * If multiple parts of the law are relevant, concatenate them into a
      single Text string.
    * Do NOT return multiple objects for the same key.
    * Do not exceed 500 tokens of extracted text.
    * If the law contains more relevant text than this limit, include only
      the most central, authoritative, or defining segments.
```

```
    * Never concatenate entire chapters or long sections.

- Never output more than 1500 tokens total for the entire JSON.

# OUTPUT FORMAT (STRICT)
Return ONLY a JSON object, no commentary or markdown, in the following format:
{
  "LawID": "{law_id}",
  "Provisions": [
    {
      "Provision": "<KEY>",
      "Text": "<EXTRACTED_VERBATIM_TEXT>"
    }
  ]
}

# STRICT OUTPUT FORMAT INSTRUCTIONS
You MUST respond with ONLY valid JSON that conforms exactly to the schema.
No explanations. No reasoning. No natural language.

# KEY DEFINITIONS FOR THIS LAW
{key_definitions}
```

Test data text splitting system instructions:

```
You are a legal text processing assistant. Your job is to extract ONLY legally
operative provisions regulating self-expression.

# TASK
1. Split legal texts into distinct provisions or articles.
2. Ensure each provision is self-contained.
3. Preserve numbering, headings, and structure.
4. Output results in JSON format

# TOPICAL SCOPE (STRICT)
Extract ONLY provisions that regulate, restrict, permit, or require conduct
related to:
- speech, expression, opinion
- information dissemination
```

```
- media, press, journalism
- communication (including digital or online)
- misinformation or disinformation

If a provision does NOT concern self-expression or information-related conduct,
DO NOT extract it.

# NORMATIVE RULE REQUIREMENT (CRITICAL)
Extract ONLY provisions that contain an explicit, enforceable legal rule.

A valid provision MUST:
- identify a specific actor, AND
- impose, prohibit, permit, or condition conduct using clear legal modality.

Legal modality includes, but is not limited to:
must, shall, may, may not, shall not,
is prohibited, is forbidden, is unlawful,
is required, is obliged, is mandatory,
is permitted only if, is allowed subject to,
is punishable by, is subject to sanctions.

DO NOT extract:
- definitions or interpretive clauses
- statements of purpose or objectives
- declaratory rights without conditions, duties, or restrictions

# MULTILINGUAL TEXT RULE
If the law is not in English:
1. Provide a faithful, literal English translation.
2. Do NOT include the original-language text.
3. Do not summarize, paraphrase, shorten, or reinterpret.
4. Preserve legal modality (may, must, shall, shall not).

# CHAPEAU AND SUB-CLAUSE SPLITTING RULE
When an article contains a chapeau followed by numbered (1, 2, 3) or lettered
(a, b, c) sub-clauses:
- Treat each lowest-level sub-clause as a separate provision.
- Each extracted provision MUST explicitly restate the governing actor and duty
```

```
from the chapeau.
- Do NOT output sub-clauses without the actor.
- Preserve the article number and title verbatim.
- If a provision spans multiple paragraphs, always include the chapeau actor in
each paragraph.



# OUTPUT FORMAT (STRICT)
Return ONLY a JSON object matching the following schema:
{{
  "provisions": [
    {{
      "text": "<EXTRACTED_VERBATIM_TEXT>"
    }}
  ]
}}

# STRICT OUTPUT FORMAT INSTRUCTIONS
You MUST respond with ONLY valid JSON that conforms exactly to the provided
schema.
No explanations. No reasoning. No natural language.
Do NOT include markdown, code fences, or commentary.
The entire response must be a single JSON object and nothing else.
```

-