QanounLLM-1o: A Large Language Model for Article-Grounded Saudi Legal Reasoning with High-Precision Citation

Muhammad Usama Saleem

University of North Carolina

Abstract

Large language models (LLMs) are increasingly used for question answering and decision support, yet legal deployment remains constrained by hallucinated claims, unreliable statutory citation, and weak jurisdictional grounding—failure modes that are particularly consequential in Saudi Arabian law, where answers must align with formally drafted Arabic instruments and their hierarchical structure. Generic or lightly adapted LLMs are ill-suited to this setting because they tend to conflate legal authorities, mis-handle document structure (e.g., articles, clauses, and nested sub-clauses), and produce fluent outputs that are not faithfully attributable to governing text. ***QanoonLLM-1o*** *is a methodology and data-centric framework for constructing Saudi-law–grounded legal assistants through (i) schema-constrained parsing of raw legal instruments into a fault-tolerant structured corpus that preserves statutory hierarchy while isolating auxiliary material, (ii) prompt-engineered synthetic supervision that is explicitly article-grounded and spans factual queries, consultation-style multi-turn interactions, and ambiguity-driven cases that require clarification rather than speculation, and (iii) a dual-layer validation pipeline that combines deterministic rule-based checks with document-conditioned LLM verification to enforce citation existence, grounding, and professional legal register.* ***QanoonLLM-1o*** *establishes an auditable foundation for training and evaluating jurisdiction-specific legal reasoning systems under strict citation and faithfulness requirements.*

1. Introduction

Large language models (LLMs) have become a dominant paradigm for natural language understanding and generation, driven by transformer architectures [10] and large-scale pretraining [1]. Instruction tuning and human-feedback optimization further improve usability in interactive settings [8]. Despite these advances, deploying LLMs for legal reasoning remains challenging, particularly in jurisdiction-specific contexts where validity hinges on statutory grounding, citation precision, and adherence to local interpretive norms. In such settings, answers that are linguistically fluent yet legally unsupported may be operationally misleading, motivating methods that explicitly prioritize legal correctness over surface-level plausibility.

Legal NLP research has explored domain adaptation through continued pretraining and benchmark-driven evaluation. Domain-specific encoders such as LegalBERT demonstrate that legal corpora improve performance on downstream legal tasks [2], while benchmark suites such as LexGLUE provide standardized evaluation across multiple legal understanding problems [3]. However, much of this literature focuses on English-centric corpora and label-based tasks, and does not directly address a core requirement of statutory question answering: identifying the applicable governing text and citing the correct article-level provisions. In parallel, retrieval-augmented generation frameworks can improve factual coverage by conditioning generation on retrieved passages [5], but retrieval alone does not guarantee that a response attributes the correct legal authority, selects the correct statutory articles, or avoids jurisdictional leakage when domain coverage is limited.

These challenges in legal documents exhibit hierarchical structure (articles, clauses, sub-clauses, exceptions), formal Arabic drafting conventions, and jurisdiction-specific terminology that are sparsely represented in general-purpose pretraining corpora. Consequently, general LLMs may hallucinate article numbers, misattribute governing authority, or rely on generic legal reasoning that is not aligned with the statutory text. Compounding this issue, conventional evaluation metrics such as n-gram overlap (e.g., BLEU) [9] can reward responses that are lexically similar to a reference while remaining legally incorrect, and broader work on truthfulness highlights that fluent generations may systematically deviate from verifiable facts

[6]. Reliable progress in jurisdiction-specific legal QA therefore requires both (i) data and modeling pipelines that enforce article-level grounding and (ii) evaluation methodologies aligned with statutory correctness rather than surface similarity.

This work introduces *QanoonLLM*, a Saudi-law–specialized legal language model designed to produce article-exact, jurisdiction-faithful legal reasoning suitable for professional use. QanoonLLM is built on a structured representation of Saudi legal documents, a curated dataset of realistic legal queries, and a supervised fine-tuning strategy that constrains generation to authoritative legal texts. To evaluate such systems, this work further proposes a layered evaluation framework that separates semantic adequacy from legal correctness by combining deterministic citation-based scoring with a structured expert-style rubric assessing legal validity and professional usability.

Our contributions are summarized as follows:

- A jurisdiction-specific legal QA pipeline for Saudi law that integrates LLM-based parsing of Arabic legal documents, structured corpus construction, realistic dataset generation, and supervised fine-tuning grounded exclusively in authoritative legal texts.
- A layered evaluation methodology tailored to statutory reasoning, combining deterministic article-level and authority-level scorers with a structured expert-style rubric that measures legal correctness beyond surface-level semantic similarity.
- A principled framework for analyzing legal LLM behavior and failure modes in high-stakes, jurisdiction-constrained environments, supporting future benchmark-driven and comparative evaluation.

2. Related Works

2.1. Legal Language Modeling and Benchmarks

A substantial body of work has investigated adapting pretrained language models to legal text, motivated by the domain's specialized vocabulary, long-range dependencies, and formal drafting conventions. Early domain-specific models such as LegalBERT demonstrate that continued pretraining on legal corpora improves performance on downstream legal NLP tasks compared to general-purpose encoders [2]. To standardize evaluation, benchmarks such as LexGLUE curate diverse legal understanding tasks (e.g., classification, retrieval, and entailment) to support reproducible comparisons across models and training regimes [3]. While these resources advance legal language understanding, most benchmark tasks do not explicitly enforce statute-level grounding (e.g., correct article identification) or authority attribution, which are central requirements for jurisdiction-constrained legal question answering.

2.2. Retrieval-Augmented and Instruction-Tuned Question Answering

Retrieval-augmented generation (RAG) combines parametric modeling with non-parametric access to external corpora to improve factual recall and reduce unsupported generation [5]. In parallel, instruction tuning and preference optimization have been shown to improve alignment with user intent and response helpfulness in open-ended question answering settings [8]. Prompting strategies that elicit intermediate reasoning, including chain-of-thought prompting, further enhance multi-step inference in large models [11]. Despite these advances, legal QA introduces additional constraints that are not directly addressed by generic QA frameworks: correct answers must be grounded in the appropriate governing text and must cite the relevant statutory articles. Moreover, retrieval and instruction tuning alone do not guarantee citation fidelity or prevent cross-jurisdictional leakage when the underlying domain is underrepresented in pretraining data.

2.3. Legal QA Data Construction and Synthetic Supervision

The availability of high-quality datasets strongly shapes model behavior in legal settings. Expert-annotated resources such as CUAD provide supervision for contract understanding and review, supporting contract-specific reasoning and extraction [4]. However, many legal QA scenarios require mapping user questions to precise statutory provisions and generating professional-grade analyses with explicit citations, which are expensive to annotate at scale. As a result, synthetic data generation has become increasingly common in instruction-tuning pipelines, but it introduces risks of hallucinated authority, imprecise citations, and stylistic artifacts that can propagate into fine-tuned models. These challenges motivate pipeline designs that emphasize structured document representations and post-generation validation to maintain legal grounding.

2.4. Evaluating Factuality, Faithfulness, and Hallucination

Standard n-gram overlap and embedding-based similarity metrics are weak proxies for correctness in high-stakes domains, as they reward surface similarity even when a response contains unsupported claims or incorrect references. Research on truthfulness and hallucination highlights that fluent generations can systematically deviate from verifiable facts [6]. More recently, black-box detection methods such as SelfCheckGPT estimate hallucination risk by assessing consistency across

sampled generations [7]. While these approaches provide useful signals for general factuality, jurisdiction-specific legal QA requires more targeted evaluation primitives that reflect statutory structure, including article-level matching and authority attribution. This motivates evaluation frameworks that explicitly separate semantic adequacy from legally grounded correctness.

3. Existing Solution: QanoonLLM-0o

This work originated from an initial baseline system, referred to as *QanoonLLM-0o*, developed as a proof-of-concept for Saudi-law–oriented legal question answering. The primary objective of this baseline was to test whether large language models could be adapted to produce synthetic question–answer supervision grounded in Saudi legal instruments. While QanoonLLM-0o established feasibility at small scale, systematic experimentation revealed architectural and methodological limitations that constrained reliability, scalability, and alignment with realistic legal use, motivating a comprehensive redesign of the end-to-end pipeline.

Dataset Generation Pipeline. QanoonLLM-0o relied on a prompt-driven dataset generation pipeline built around four manually engineered prompt templates. These templates were used to synthesize question–answer pairs from parsed Saudi legal documents. The pipeline implicitly assumed that the relevant legal document was available to the model at inference time, and the generated answers predominantly followed a summary-oriented style that paraphrased the source text rather than providing precise, article-grounded legal reasoning.

Hard-Coded Parsing and Validation. QanoonLLM-0o implemented parsing and validation using hard-coded, rule-based logic. A regex-driven parser extracted document titles, article headers, and a limited set of structural markers from Arabic legal text, and downstream validation relied on deterministic checks tightly coupled to these extracted patterns. This design imposed strong assumptions about document formatting and offered limited extensibility: accommodating a new drafting convention typically required manual rule updates. As a result, the system was brittle under real-world variability. Minor shifts in layout (e.g., line breaks, punctuation), numbering conventions, or clause/sub-clause nesting frequently produced incomplete or incorrect parses. Downstream dataset generation then operated on partial or malformed representations, leading to truncated provisions, mis-segmented hierarchy, and inconsistent supervision. The validation stage exhibited analogous limitations. Fixed rules could reliably detect only a narrow class of surface-form issues, but lacked semantic awareness needed to identify hallucinated claims, non-existent or incorrect citations, or subtle mismatches between an answer and the governing text. Consequently, QanoonLLM-0o provided no principled mechanism to enforce citation existence and document-grounded faithfulness at scale.

Limitations of QanoonLLM-0o. Through iterative analysis, we identified several key limitations in the QanoonLLM-0o baseline:

- **Fragile parsing and validation**: Hard-coded parsers and validators were highly sensitive to variations in Saudi legal drafting (e.g., numbering schemes, nested sub-clauses, and secondary sections), leading to frequent extraction failures and malformed intermediate structures.
- **Summary-oriented supervision**: Generated answers often prioritized paraphrased summaries over clause-faithful, article-grounded reasoning with explicit statutory traceability.
- **Implicit document availability**: Many examples referenced "the provided document" or "the text above," modeling a context-injected setting rather than a realistic deployment scenario requiring explicit citations anchored to a structured legal source.
- **Questions sounded generated rather than human-asked**: Many questions were written in a rigid, "textbook" style instead of sounding like how real clients or lawyers ask. For example, questions often looked like: "Explain Article X in detail and list all definitions and penalties," rather than natural requests such as "Does this rule apply to my situation?" or "What do I need to do to comply?" As a result, the dataset contained fewer realistic scenario-based questions and fewer back-and-forth clarification questions, which can make the trained assistant less aligned with real consultation conversations.
- **Missing or inconsistent article number & law title usage in citations**: Some answers referenced article numbers without consistently including the correct instrument name/title, producing incomplete legal references and reducing citation reliability for downstream training and auditing.
- **Arabic text leakage into English datasets**: In mixed-language generation settings, some English prompts and answers contained Arabic terms or code-switched spans, breaking linguistic consistency and introducing distributional noise that can degrade downstream model behavior and evaluation.

- **Incomplete fine-grained coverage**: Targeted supervision at clause/sub-clause resolution was inconsistent, limiting exposure to fine-grained citation behavior and hierarchical reasoning.
- **No ambiguity-driven dialogues**: The pipeline did not systematically generate clarification-focused interactions for missing, conflicting, or incorrectly specified context, reducing robustness to common real-user failure modes.
- **Inefficient generation runtime**: Redundant LLM calls and limited reuse of intermediate artifacts increased cost and latency, constraining scaling across additional instruments.
- **Insufficient data quality assurance**: Generated samples were not evaluated with semantic grounding checks, allowing uncited claims, incorrect references, and weakly supported reasoning to propagate into fine-tuning data.

Taken together, these limitations exposed a gap between fluent language generation and legally reliable, citation-faithful reasoning. Addressing this gap required rethinking not only prompt design, but the overall system architecture, including document parsing, supervision construction, validation methodology, and dataset composition to better reflect deployment-critical behaviors. The remainder of this paper presents the redesigned QanoonLLM framework, which directly targets these shortcomings through schema-constrained parsing, article-grounded synthetic supervision, and dual-layer validation.

4. Proposed Method: QanoonLLM

Problem Formulation. Given a user query concerning an actual or hypothetical legal dispute, the objective is to generate a precise, jurisdiction-consistent legal response grounded exclusively in Saudi laws and regulations. To be operationally viable for professional legal use, the system must satisfy four core requirements: (i) correctly identify and cite the relevant statutory articles; (ii) accurately summarize the legal obligations, prohibitions, or rights established by those provisions; (iii) explicitly state any penalties or sanctions prescribed therein; and (iv) reason in accordance with recognized Saudi legal interpretive practices.

General-purpose large language models are not optimized for this setting. They lack systematic exposure to Saudi legal terminology, Arabic legislative structure, and jurisdiction-specific normative reasoning, which frequently results in speculative interpretations, incorrect citations, or cross-jurisdictional leakage. These limitations make such models unsuitable for high-stakes legal applications. A domain-adapted approach is therefore required to ensure article-level precision, jurisdictional fidelity, and legally defensible outputs. QanoonLLM is designed to address these requirements directly.

Overview of the Proposed Method. QanoonLLM is a Saudi-law–specialized legal language model engineered to produce article-grounded, sanction-aware legal analyses. The proposed method follows a two-stage pipeline that couples structured corpus construction with supervised fine-tuning tailored to statutory reasoning.

The first stage focuses on **corpus construction and structuring**. Authoritative Saudi legal sources—including statutes, regulations, executive bylaws, and enforcement instruments—are systematically collected and consolidated into a unified corpus. These texts are cleaned, normalized, and segmented into legally coherent units such as articles, clauses, sub-clauses, exceptions, and sanctions. This structured representation preserves the hierarchical organization of Saudi legislation and enables the generation of high-fidelity supervised question–answer pairs that reflect realistic legal reasoning tasks rather than surface-level summarization.

The second stage involves **fine-tuning the QanoonLLM model** on the curated dataset. Fine-tuning is performed to learn robust mappings between user queries or scenarios and the correct statutory provisions, with explicit emphasis on article citation accuracy, bilingual Arabic–English comprehension, and structured legal reasoning. This training regime is designed to minimize hallucination, enforce jurisdictional constraints, and promote consistent interpretive behavior across varying levels of legal complexity. Together, these stages yield a model capable of delivering reliable, article-exact legal reasoning suitable for professional Saudi legal contexts.

4.1. Data Pre-processing

4.1.1. Problem Setting and Design Requirements

Structural Variability in Saudi Legal Texts. The preprocessing pipeline operates on raw Saudi legal documents provided as unstructured text. These documents vary substantially across instrument types (e.g., decrees, council decisions, and implementing regulations) and across publications within the same type. The source text often interleaves headers, decree metadata, extended preambles, decision lists, and statutory articles. Article bodies may contain multi-level hierarchy (clauses and sub-clauses) expressed through heterogeneous conventions, including ordinal-based headings, lettered sub-items, and nested numbering patterns such as “1/1”, “2/1”, and “3/1”. Many documents further append supplementary material (e.g., appendices, schedules, and tables) after the statutory body, which must be isolated to prevent contaminating article boundaries and downstream citation logic.

Motivation for Contextual Parsing. Pattern-based parsing is brittle under this variability: small shifts in punctuation, numeral styles, line breaks, or heading conventions can break extraction rules or introduce hierarchy errors. These failures are particularly harmful for downstream learning, where dataset generation and supervised fine-tuning assume a faithful article- and clause-level segmentation. The preprocessing pipeline is therefore designed to (i) resolve hierarchy using contextual interpretation, (ii) enforce a schema-consistent representation, and (iii) remain conservative when structure is ambiguous.

4.1.2. Input Processing and Minimal Normalization

Raw Input Format. Each document is ingested as a UTF-8 text string. The pipeline preserves the original ordering of lines and sections and treats the raw text as the sole source of authority.

Minimal Text Normalization. To reduce superficial noise without altering legal meaning, preprocessing applies lightweight normalization: whitespace is canonicalized and obvious encoding artifacts are removed. No heuristic rewriting, paraphrasing, or restructuring of statutory language is performed prior to parsing.

4.1.3. Schema-Constrained LLM Parsing

Structured Output Contract. The central transformation maps raw text to a schema-consistent JSON representation, which is subsequently materialized into a typed document object. The schema includes fixed keys for: document titles (Arabic and English), decree metadata (number and date), issuing authority, preamble, decisions, definitions, articles (with nested clauses and sub-clauses), appendices, an optional secondary section descriptor for council decisions, and metadata (e.g., Hijri dates, cross-document references, and total article count). This representation is designed to be deterministically consumable by downstream dataset generation, validation, and citation formatting.

Prompt Contract and Faithfulness Constraints. Parsing is performed by prompting an LLM with (i) a system role that specifies expert legal document parsing behavior and (ii) an explicit schema contract that enumerates required keys, field types, nesting structure, and permissible empty values. The prompt enforces strict faithfulness constraints: the model is instructed to extract only content explicitly present in the source text and to leave fields empty when uncertain. To preserve drafting signals needed for hierarchy reconstruction, the full raw document text is embedded in the prompt without reordering.

Structure-Sensitive Extraction Rules. The prompt explicitly constrains ambiguity-prone hierarchy markers commonly observed in these documents. In particular, it requires: (i) preserving ordinal phrases verbatim when they appear in article headings; (ii) treating decision ordinals (e.g., "Firstly", "Secondly") as independent decision or clause units rather than introducing implied nesting; (iii) extracting definition lists introduced by a colon as article-contained clauses even when not explicitly numbered; (iv) treating lettered items (e.g., "a–d") as sub-clauses under a parent clause rather than separate clauses; and (v) interpreting nested numbering patterns ("X/Y") as sub-clauses under the corresponding parent clause ("Y-") while preserving the full pattern as an identifier and storing only the trailing text as sub-clause content.

4.1.4. Fault-Tolerant Parsing Workflow

Deterministic Multi-Stage Workflow. Parsing follows a deterministic workflow that guarantees a valid output object under LLM instability. First, a minimal fallback document is constructed from the raw text. Second, the schema-constrained prompt is assembled and sent to the LLM via a JSON-returning interface. Third, the pipeline validates and safely converts the returned JSON into a typed representation, applying conservative defaults for missing or malformed fields.

Retry Policy and Failure Recovery. LLM parsing is attempted up to a fixed number of retries (default: 3) with a constant delay between attempts (default: 2 seconds). Retries are triggered by transient API failures, invalid JSON, null responses, or structurally empty outputs (e.g., missing both a title and extracted articles). If all retries fail, preprocessing returns the minimal fallback document, ensuring the pipeline never raises due to parsing failures and never emits null outputs.

4.1.5. Response Validation and Safe Document Construction

Schema and Type Validation. Each LLM response undergoes validation before acceptance, including JSON well-formedness, schema-level checks (required keys and correct nesting), and type checks for all fields. When a field is present but ill-typed (e.g., decisions not a list, definitions not a dictionary), the pipeline falls back to the corresponding fallback value rather than attempting inference.

Minimal Content Checks. To prevent propagating degenerate parses, the pipeline requires minimal evidence of structure: either at least one extracted article or a non-trivial title that differs from the fallback heuristic. If these checks fail, the attempt is treated as unsuccessful and triggers a retry when available.

Conservative Field Repair. When partially valid outputs are accepted, missing or malformed fields are conservatively repaired using empty defaults (empty strings, empty lists, empty dictionaries). This avoids introducing spurious structure into downstream dataset generation and prevents silent schema drift.

4.1.6. Hierarchy Interpretation and Canonicalization

Articles, Clauses, and Sub-Clauses. The structured representation explicitly models multi-level hierarchy. Articles include free-form content and an optional list of clauses. Each clause contains content and an optional list of sub-clauses. Sub-clauses encode a numeric index and an identifier: for lettered sub-clauses, the identifier corresponds to the letter marker; for nested numbering patterns, the identifier stores the full "X/Y" pattern, while the sub-clause content stores only the trailing text following that marker. Sub-clause indices are enforced to restart from 1 for each new parent clause.

Decision vs. Article Disambiguation. The parser separates decision lists from statutory articles using contextual cues and ordinal usage. Ordinal markers and numbered items that are peers in drafting are represented as siblings rather than parent–child relations (e.g., an ordinal heading followed by "1-", "2-" items), preventing erroneous nesting that would distort the legal hierarchy.

4.1.7. Post-Parsing Normalization and Quality Controls

Decision Deduplication. Decisions may be duplicated across sections, particularly when a decree and a secondary council decision section restate materially identical directives. To avoid duplicating training context, the pipeline deduplicates decisions after parsing when enabled. Decision text is normalized (collapsed whitespace and punctuation normalization) and compared using sequence-based similarity. A decision signature combines its ordinal (if present) with a configurable prefix of normalized content (default: 200 characters). Decisions are removed when similarity exceeds a configurable threshold (default: 0.90), with a stricter threshold applied when ordinals are absent to reduce false positives.

Metadata Extraction with Dual Strategy. The pipeline populates metadata used for downstream conditioning and citation formatting. If the LLM returns non-empty metadata fields (dates or references), these are used directly; otherwise, metadata is extracted using regex-based fallbacks. Date extraction targets full Hijri date formats (day/month/year) as well as year-only mentions and supports multiple numeral representations. Reference extraction targets canonical cross-document citations (e.g., decision numbers and dates, decree references). In all cases, the total article count is set deterministically from the constructed article list.

4.1.8. Supplementary Materials and Section-Specific Handling

Appendices Isolation. Supplementary materials (e.g., fee schedules and violation tables) frequently follow the main statutory body and can introduce spurious boundaries if treated as article text. The pipeline therefore extracts such content into a dedicated appendices field. The parsing prompt explicitly instructs the model not to merge appendices into the final article, preserving article boundaries for downstream dataset generation.

Secondary Section Extraction for Council Decisions. Some documents contain a distinct secondary section corresponding to council decisions, with its own header and preamble. The pipeline extracts this header/preamble into a dedicated field while placing the enumerated decisions into the unified decisions list. Decisions from both the primary and secondary sections are included prior to deduplication, preserving section provenance while enabling normalization of repeated content.

Overall, the preprocessing pipeline converts raw legal text into a schema-consistent and fault-tolerant structured representation. The design emphasizes conservative handling of ambiguity, explicit validation, deterministic fallback behavior, and post-parsing normalization, ensuring that downstream dataset generation and supervised fine-tuning operate on stable article- and clause-level structure.

4.2. Synthetic Dataset Generation

This section describes the prompt-engineered synthetic data pipeline used to construct supervision for Saudi legal question answering and consultation-style interactions. The pipeline focuses on generating article-grounded question–answer pairs and multi-turn dialogues that reflect professional legal practice, while enforcing conservative behavior under ambiguity. This section covers data generation only; model training and evaluation are discussed separately.

4.2.1. Objectives and Scope

The primary objective is to produce synthetic supervision suitable for fine-tuning language models to operate as a jurisdiction-constrained assistant for Saudi laws and regulations. The generated data is designed to (i) map realistic user queries to the

relevant statutory provisions, (ii) elicit professionally appropriate explanations of obligations, prohibitions, rights, and sanctions when present in the source, and (iii) model responsible interaction behaviors such as clarification and non-speculative responses. The scope of this component is restricted to interaction design and prompt construction for data generation, excluding model optimization and benchmarking.

4.2.2. Data Design Principles

The dataset construction is governed by four explicit principles that directly affect downstream learning behavior.

Legal realism. Prompts are designed to approximate how legal questions arise in practice, including short fact patterns, procedural queries, and requests for article interpretation. This reduces reliance on exam-style phrasing that can bias models toward rote paraphrase rather than consultative assistance.

Strict grounding in authoritative sources. Each generated example is constrained to the statutory text provided as context. The assistant is instructed to cite the governing legal instrument and relevant articles and to avoid introducing legal claims not supported by the text. This grounding constraint is critical for jurisdiction-specific deployment, where plausible but ungrounded answers constitute a primary failure mode.

Diversity and coverage. The prompt set is designed to vary question intent (definition, scope, obligation, exception, penalty, authority), interaction format (single-turn vs. multi-turn), and reference style (article-explicit vs. law-level). This diversity encourages broad coverage of statutory behaviors and reduces overfitting to a narrow interaction template.

Controlled ambiguity and conservative behavior. Real-world legal queries often omit key facts or reference the wrong authority. The generation pipeline therefore includes prompts that deliberately under-specify context or introduce mismatched assumptions. The assistant is instructed to request clarification, state uncertainty explicitly, and avoid definitive conclusions when the statutory basis cannot be determined from the provided text.

4.2.3. Prompt Engineering Strategy

Prompt engineering is treated as a control mechanism for producing consistent, schema-aligned interactions rather than ad-hoc instruction. Each generation prompt specifies (i) the interaction mode and required output structure, (ii) the permissible scope of the answer (restricted to the provided statutory text), and (iii) citation and grounding requirements (governing law name and article-level references). Prompts are parameterized by the target unit of supervision (law-level, article-level, or scenario-level) and by a complexity level (simple/medium/complex). This design enables systematic coverage of interaction behaviors while keeping outputs compatible with downstream validation and filtering.

4.2.4. Prompt Taxonomy

To reflect the range of professional legal tasks, the pipeline uses multiple prompt categories, each modeling a distinct interaction pattern.

Single-turn question–answer prompts. These prompts generate self-contained question–answer pairs with explicit constraints on citation behavior and answer scope.

- **Direct factual prompts** elicit concise answers about a specific rule, obligation, condition, or institutional authority, with mandatory article citation when applicable.
- **Law-level scope prompts** ask what a legal instrument governs (purpose, coverage, regulated activity) without requiring an article reference, encouraging high-level but grounded descriptions.
- **Article-exegesis prompts** explicitly cite an article and request what it provides or establishes, requiring a complete, article-contained explanation rather than a partial paraphrase.
- **Article-effect prompts** cite an article and ask about its legal effect (scope of power, applicability, consequences), requiring bounded interpretive explanation grounded in the article text.

Multi-turn consultation prompts. These prompts generate dialogues in which the user and assistant iteratively refine the legal question, resembling client–advisor interactions.

- **Standard consultations** assume the governing instrument is identifiable from the initial user query and progress through limited follow-up to a grounded conclusion.

- **Scenario-based consultations** present a fact pattern and require the assistant to map facts to statutory provisions, explicitly stating conditions, exceptions, and procedural steps that are supported by the text.
- **Ambiguity-driven consultations** intentionally omit key facts or introduce incorrect statutory references, requiring the assistant to request clarification, propose candidate provisions cautiously, or revise the answer after user correction.

Reasoning-oriented prompts. These prompts target tasks that require controlled synthesis across multiple provisions or procedural elements (e.g., multi-condition compliance, exception handling, or authority boundaries). The output is constrained to structured reasoning grounded in cited provisions, with explicit instructions to avoid speculation and to surface missing facts as clarification requests.

4.2.5. Prompt Complexity Control

Complexity is treated as an independent axis applied across all prompt types, enabling progression from factual recall to applied statutory reasoning while preserving consistent output expectations.

Simple. Simple prompts focus on isolated provisions or clearly bounded concepts (definitions, direct obligations, scope statements) that can be answered with minimal inference. They primarily train accurate retrieval and faithful restatement at the article level.

Medium. Medium prompts introduce conditionality or limited cross-referencing (e.g., prerequisites, institutional roles, or procedural requirements) within a clearly identifiable legal framework. These prompts require structured application rather than direct recall, but do not depend on resolving substantial ambiguity.

Complex. Complex prompts require deeper synthesis across multiple provisions, layered conditions, and exception handling, and may be instantiated in single-turn or multi-turn form. Complexity is introduced through interactions between statutory rules, procedural sequencing, or evolving information across dialogue turns. The assistant is constrained to articulate assumptions explicitly, request missing information when necessary, and ground every substantive claim in cited provisions.

4.2.6. Ambiguity Modeling and Responsible Interaction

Ambiguity is modeled explicitly to train responsible advisory behavior under incomplete information. Prompts introduce ambiguity by withholding facts required for applicability, using underspecified references (e.g., naming a topic but not the instrument), or including incorrect assumptions about the governing authority. The assistant is instructed to (i) ask targeted clarification questions when the statutory basis cannot be determined, (ii) avoid definitive conclusions without textual support, and (iii) revise the response when the user supplies corrections or additional context. This interaction design aims to reduce overconfident, ungrounded outputs and align generation behavior with professional expectations for jurisdiction-specific legal consultation.

4.3. Validation

Motivation and Validation Objectives. Synthetic supervision generated by large language models can contain failure modes that are difficult to detect with surface-level checks, including hallucinated statutory claims, missing or incorrect citations, and informal or opinionated phrasing that is inappropriate in legal settings. These risks are amplified in jurisdiction-constrained legal assistance, where a fluent but uncited or mis-cited answer can be materially misleading. The validation framework is therefore designed to enforce (i) grounding in the provided source document, (ii) citation presence and correctness, (iii) professional and objective language, and (iv) consistent, auditable outputs suitable for downstream fine-tuning.

Validation Methodology Overview. Validation follows a dual-layer paradigm that combines deterministic rule-based checks with an LLM-based semantic validator. The rule-based layer provides fast, reproducible detection of citation patterns and prohibited language, while the LLM-based layer performs document-conditioned verification of factual claims and identifies subtle mis-grounding that may evade pattern matching. The system adopts conservative acceptance: examples are considered valid only when they satisfy both validation pathways and meet a minimum quantitative score.

Framework Architecture and Responsibilities. The framework operates on generated examples paired with their originating source document. The source document is first parsed into a structured representation that enumerates available articles

and other addressable units (e.g., decision items where applicable). Validation then proceeds in two complementary pathways: (i) a rule-based validator that extracts and analyzes observable patterns in the generated text (citations, suspicious hedging phrases, forbidden opinionated language), and (ii) an LLM-based validator that receives the full conversation and the source document context and returns a structured report of grounding, citation plausibility, and unsupported claims. Both pathways emit structured diagnostics (errors, warnings, and per-criterion metadata) and a normalized score in $[0, 1]$.

Hallucination and Grounding Validation. Grounding validation targets the core requirement that all substantive legal claims must be attributable to the provided source document. The validation criteria include: (i) **citation presence**, requiring at least one explicit statutory citation when citations are mandated; (ii) **factual grounding**, flagging claims that introduce requirements, procedures, dates, numbers, or legal effects not supported by the source text; (iii) **suspicious/hedging language**, which is treated as a warning signal because it often correlates with speculative completion rather than direct grounding; and (iv) **forbidden opinionated language**, which is treated as a critical failure because legal advisory content must remain objective and source-based. Violations are separated into *critical errors* (invalidate the example regardless of score) and *warnings* (reduce confidence and lower the quantitative score but do not necessarily invalidate on their own).

Citation Validation Against the Parsed Document. Citation validation ensures that references in generated content are both syntactically well-formed and semantically resolvable against the parsed document structure. The validator first extracts all citation mentions from the generated response using language-appropriate templates (e.g., "According to Article X" patterns in English and their Arabic counterparts), including article-only and article-plus-paragraph variants when present. Each extracted reference is then cross-checked against the document's parsed inventory of articles to verify existence. Citations that refer to non-existent provisions are marked invalid and treated as critical failures. In addition, the validator checks citation sufficiency: responses that contain multiple substantive claims but provide fewer than the required minimum number of citations are penalized, and in strict settings may be rejected.

Quality and Linguistic Assessment. In addition to grounding and citation correctness, the framework enforces professional legal communication constraints. The validator checks for: (i) **formal register** (avoidance of colloquialisms and informal phrasing), (ii) **objective tone** (absence of personal opinions and subjective assertions), (iii) **terminological consistency** (stable use of legal terms and roles across an answer), and (iv) **structural coherence** (answers must remain logically organized and avoid contradictory statements across turns). These criteria are assessed through a combination of deterministic pattern checks (for forbidden phrases) and LLM-based semantic assessment for nuanced issues (e.g., subtle misinterpretations or internally inconsistent reasoning).

Quantitative Scoring and Acceptance Criteria. Each example is assigned a deduction-based score in $[0, 1]$. All examples start with a base score of 1.0 and incur penalties for detected issues: missing citations (-0.3), suspicious hedging phrases (-0.1 per instance, capped at -0.3), forbidden opinionated language (-0.2 per instance), major factual inaccuracies (-0.2 per instance), unsupported claims (-0.15 per claim), and additional general errors (-0.1 per error). The final score is clamped to $[0, 1]$. An example is classified as valid only if (i) it contains no critical errors (e.g., invalid citations or forbidden opinionated language), and (ii) it meets a minimum score threshold (default: ≥ 0.8). This criterion is applied conservatively across both validation pathways.

Dual Validation Pathways and Conservative Combination. The rule-based pathway provides deterministic detection of citation presence, citation pattern matches, and prohibited language markers, and is used to enforce hard constraints efficiently. The LLM-based pathway performs document-conditioned semantic verification, checking whether the assistant response is supported by the source document and whether any content appears invented or materially misaligned. The two pathways operate independently and their outputs are combined conservatively: an example must pass both validators to be retained. When validators disagree, the framework defaults to rejection, storing the full diagnostics for analysis and iterative improvement of generation prompts.

End-to-End Validation Workflow. Validation proceeds as a structured pipeline over generated examples grouped by source document: (i) the system loads generated examples and the corresponding source document, (ii) the source document is parsed into a structured representation to enable citation existence checks, (iii) examples are normalized into a standardized internal format that preserves the full conversation and metadata (e.g., language, prompt type, complexity), (iv) each example is validated sequentially by the rule-based and LLM-based validators, (v) scores, errors, and warnings are aggregated into a unified validation report, and (vi) examples are classified as accepted or rejected based on the conservative acceptance criteria. The pipeline is designed to be fault-tolerant: failures in the semantic validator (e.g., transient LLM errors) do not terminate processing, and affected examples are conservatively marked as invalid with an explicit failure reason.

Validation Outputs and Diagnostics. For each processed example, the framework produces a structured validation record containing: binary validity, final score, categorized errors and warnings, extracted citations with validity labels, and a detailed

per-criterion report from the semantic validator. Accepted and rejected examples are stored separately while preserving the original example content and appending validation metadata. Aggregate summaries (counts, pass rate, and score distributions by prompt category and complexity) are produced to support auditing, to identify dominant failure modes (e.g., missing citations vs. invalid references), and to guide iterative refinement of the generation prompts and constraints.

4.4. Finetuning

Fine-tuning is performed using the **OpenAI** fine-tuning pipeline with **GPT-4o** as the base model. Training examples are serialized in a chat-style format consistent with the generation outputs (single-turn QA, multi-turn consultations, and reasoning-oriented interactions), and each example preserves article-grounded citation patterns to maintain provenance constraints during adaptation. The fine-tuning corpus is drawn from the deterministic train split (Section 5) and retains per-example metadata linking each interaction to its source document and generation category, enabling traceability and auditability throughout the training process. Where validation is enabled, only examples that satisfy citation existence and grounding checks are admitted to the fine-tuning set, ensuring that the supervision signal remains consistent with the framework's strict faithfulness requirements.

5. Implementation Details

Configuration and Reproducibility. All stages of the pipeline are governed by a single declarative configuration that specifies the selected LLM backend, parsing and generation robustness policies, dataset generation budgets, output conventions, deterministic splitting parameters, and an optional validation stage. Reproducibility is enforced through a fixed random seed (42) for all shuffling and splits, a deterministic directory convention that isolates outputs by source document, and fixed per-condition sampling budgets. Execution is logged to both console and a persistent log file to support traceability of parsing/generation outcomes and aggregate sample counts.

LLM Backend and Sampling Configuration. The system abstracts multiple LLM backends behind a uniform interface for schema-oriented prompting and JSON-constrained responses. Unless otherwise noted, the primary backend uses Gemini (`gemini-2.5-flash-preview-09-2025`) with a maximum output budget of 65,536 tokens and temperature 0.7. Alternative configurations for OpenAI (`gpt-4o`) and Claude (`claude-sonnet-4-20250514`) are supported using the same interaction contract, enabling backend substitution while preserving the data schema and generation protocol.

Data Ingestion and Intermediate Artifacts. Raw legal instruments are ingested as UTF-8 plaintext documents (`.txt`) from a document repository and processed either in single-document mode or batch mode. Each input document is converted into a schema-consistent structured representation and written as an intermediate parsed artifact. This intermediate form serves as the stable substrate for subsequent synthetic generation and (when enabled) validation, ensuring that all downstream stages operate on a consistent document structure rather than raw free-form text.

Synthetic Generation Parameters. Synthetic data generation is organized along three explicit axes: language, interaction type, and complexity. In the reported configuration, generation is performed in Arabic and spans single-turn question answering, multi-turn consultation variants, and reasoning-oriented interactions, including both generic and article-based question forms. Complexity is treated as an independent dimension with three levels (*simple*, *medium*, *complex*). For each interaction type and complexity level, the pipeline generates 20 examples, yielding uniform coverage across the predefined grid of conditions.

Output Structure and Provenance. Outputs are materialized as JSONL files and stored under a document-isolated directory structure, with one numbered subdirectory per source document and separate files per interaction type (and subtype where applicable). Alongside examples, the pipeline optionally emits metadata summaries that record document identifiers, generation conditions, and aggregate counts. This layout preserves provenance, enables per-document auditing, and supports modular re-generation or filtering without requiring recomputation of unrelated documents.

Dataset Splitting and Final Dataset Size. The final dataset is derived from 9 source legal documents. For each document, generation produces 20 samples per configured (interaction type, complexity) condition, yielding an initial pool of approximately 10,000 examples and a final partition of approximately 9,000 training examples and 1,000 test examples. The split is deterministic under the fixed seed (42). A 15% holdout is applied at the file level for evaluation-critical categories (QA, reasoning, generic QA, and article-based QA), while remaining categories are assigned to training to preserve breadth of supervision. After aggregation across documents, both training and test sets are globally shuffled and written as consolidated JSONL files.

Validation Controls. Validation is implemented as a configurable component that can be enabled or disabled without changing the generation protocol. When enabled, validation applies explicit acceptance criteria targeting source grounding, citation

correctness, factual consistency, date consistency, and a no-inference constraint, together with length bounds for questions and answers and a fixed citation template for the target language. In the reported configuration snapshot, validation is disabled, allowing generation and splitting to run independently; the configuration nonetheless defines the validation contract to ensure that acceptance criteria remain explicit and reproducible when validation is activated.

6. Evaluation Methodology

We evaluate legal question answering performance using a layered evaluation framework explicitly designed to separate surface-level semantic fluency from substantive legal correctness. The proposed methodology combines deterministic, citation-based scoring mechanisms with a structured expert-style rubric, enabling precise assessment of article-level faithfulness, correct attribution of legal authority, and the soundness of legal reasoning.

Article Scorer: Article-Level Legal Accuracy. To assess whether a model correctly cites the relevant statutory provisions, we introduce an *Article Scorer* that measures article-level legal accuracy independently of linguistic similarity. This scorer evaluates whether the article numbers referenced in the model-generated response align with those cited in the ground-truth legal answer.

Let y denote the ground-truth answer and $\hat{y}$ the model output. We define:

- A_{GT} as the set of article numbers cited in y,
- A_{M} as the set of article numbers cited in $\hat{y}$.

Article numbers are extracted using deterministic pattern matching (e.g., "Article 3", "Article 5"), with normalization applied to account for formatting and linguistic variations. The set of correctly matched articles is defined as:

$$A_{\text{match}} = A_{\text{GT}} \cap A_{\text{M}}.$$

The Article Score is computed as:

$$S_{\text{article}} = \begin{cases} \dfrac{|A_{\text{match}}|}{|A_{\text{GT}}|} \times 100, & \text{if } |A_{\text{GT}}| > 0, \\ 0, & \text{if } |A_{\text{GT}}| = 0. \end{cases}$$

This formulation assigns full credit only when all required articles are correctly cited, partial credit when only a subset is cited, and zero credit when no correct articles are referenced. Importantly, extraneous or hallucinated articles do not increase the score, ensuring that over-citation does not obscure incomplete legal grounding.

Governing Body Scorer: Legal Authority Attribution. Correct article citation alone is insufficient if the cited provisions are attributed to an incorrect legal authority. To address this limitation, we introduce a *Governing Body Scorer* that evaluates whether the model references the same governing legal texts as those specified in the ground truth.

We define:

- G_{GT} as the set of governing legal texts cited in the ground-truth answer,
- G_{M} as the set of governing legal texts cited in the model response.

Governing texts include the names of laws, regulations, or bylaws. Minor lexical or formatting variations are permitted, provided the referenced legal source is substantively identical. The set of correctly matched governing texts is defined as:

$$G_{\text{match}} = G_{\text{GT}} \cap G_{\text{M}}.$$

The Governing Body Score is computed as:

$$S_{\text{gov}} = \begin{cases} \dfrac{|G_{\text{match}}|}{|G_{\text{GT}}|} \times 100, & \text{if } |G_{\text{GT}}| > 0, \\ 0, & \text{if } |G_{\text{GT}}| = 0. \end{cases}$$

This scorer enforces correct attribution of legal authority and prevents jurisdictional errors, such as citing valid statutory articles under an unrelated or incorrect legal framework.

Evaluation Based on Ten Legal Criteria. While article- and authority-level scorers enforce citation correctness, they do not fully capture the quality of legal reasoning. We therefore introduce a structured *Ten-Criteria Legal Evaluation* rubric to assess end-to-end legal validity and professional adequacy.

Each criterion is evaluated independently and assigned a discrete score:

$$s_i \in \{0, 5, 10\}, \quad i = 1, \ldots, 10,$$

where 0 denotes failure, 5 indicates partial satisfaction, and 10 indicates full satisfaction.

The ten evaluation criteria are:

1. Correct Article Citation
2. Correct Governing Legal Text
3. Accuracy of Article Summaries
4. Absence of Hallucinated Articles
5. Completeness of Legal Citation
6. Soundness of Legal Reasoning
7. Handling of Legal Exceptions and Conditions
8. Jurisdictional Correctness
9. Professional Legal Tone and Clarity
10. Actionable and Defensible Conclusion

The final rubric score is computed as:

$$S_{\text{10-criteria}} = \sum_{i=1}^{10} s_i,$$

yielding a maximum possible score of 100.

This rubric mirrors expert legal review by independently evaluating citation accuracy, statutory interpretation, reasoning soundness, jurisdictional alignment, and professional usability. The non-overlapping structure enables fine-grained diagnostics of model behavior while preventing score inflation due to surface-level fluency.

Evaluation Design Rationale. Together, the Article Scorer, Governing Body Scorer, and Ten-Criteria rubric form a layered evaluation framework that explicitly distinguishes semantic similarity from legal correctness. Textual fluency or semantic overlap alone is insufficient for legal question answering tasks; authoritative correctness is enforced only through explicit citation matching, correct attribution of legal authority, and structured reasoning assessment. This design prevents legally fluent but incorrect answers from receiving inflated scores and ensures robustness in jurisdiction-specific, high-stakes legal applications.

7. Conclusion

This paper presented **QanoonLLM-1o**, a methodology and data-centric framework for building Saudi-law–grounded legal assistants under strict faithfulness and citation requirements. The framework integrates (i) schema-constrained parsing that converts raw Arabic legal instruments into a fault-tolerant, hierarchy-preserving structured corpus, (ii) prompt-engineered synthetic supervision that is explicitly article-grounded and spans factual QA, consultation-style multi-turn interactions, and ambiguity-driven cases that favor clarification over speculation, and (iii) a dual-layer validation pipeline that combines deterministic rule-based checks with document-conditioned LLM verification to enforce citation existence, grounding, and professional legal register. Together, these components provide an auditable pipeline for generating training-ready supervision in a setting where provenance is mandatory and structural errors are consequential.

More broadly, QanoonLLM-1o offers a reproducible template for reliability-oriented LLM development in structured regulatory domains where authoritative sources are hierarchical and citation-sensitive. Current limitations include dependence on corpus completeness and drafting variability, and the inherent mismatch between synthetic interactions and real practitioner or client behavior. Future work will expand legal coverage, improve interaction realism through iterative refinement with expert feedback, support multi-document and scenario-based reasoning with explicit provenance across sources, and strengthen evaluation protocols that separate statutory grounding and citation correctness from surface-level linguistic quality.

8. Future Work

Future work will focus on extending QanoonLLM-1o along four directions. First, the framework will adopt a standardized, machine-checkable output format for generated answers and multi-turn dialogues to further reduce schema drift and simplify downstream auditing across models and languages. Second, the data pipeline will be expanded to support end-to-end domain

adaptation, including finalizing a local (self-hosted) LLM for dataset generation and incorporating fine-tuning and evaluation as reproducible stages within the same configuration-driven workflow, alongside provider-based alternatives (e.g., Gemini-based fine-tuning) where appropriate.

Third, the framework will prioritize practitioner alignment by bridging the gap between technical development and legal practice, including structured review loops with domain experts and scenario-based interaction design that better reflects client consultations, procedural advice, and documentation requirements. Finally, future datasets will emphasize cross-instrument reasoning by introducing interactions that require referencing multiple legal documents within a single exchange, enabling controlled study of multi-source grounding, conflict resolution, and citation behavior under realistic regulatory dependencies.

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020. 1

[2] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. LEGAL-BERT: The muppets straight out of law school. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2898–2904, 2020. 1, 2

[3] Ilias Chalkidis, Abhik Jana, Dirk Hartung, Nicola Bombieri, Heiner Stuckenschmidt, Rodrigo Martin, and Lilja Øvrelid. Lexglue: A benchmark dataset for legal language understanding in English. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4310–4330, 2022. 1, 2

[4] Dan Hendrycks, Collin Burns, Steven Basart, Reed Wild, Andy Zou, Mohammadreza Mostajabi, Jacob Steinhardt, and Dawn Song. CUAD: An expert-annotated NLP dataset for legal contract review. In *Advances in Neural Information Processing Systems*, pages 24694–24706, 2021. 2

[5] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, pages 9459–9474, 2020. 1, 2

[6] Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Advances in Neural Information Processing Systems*, pages 19791–19803, 2021. 2

[7] Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9022, 2023. 3

[8] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, pages 27730–27744, 2022. 1, 2

[9] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, 2002. 1

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. 1

[11] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, pages 24824–24837, 2022. 2