From Policy to Logic for Efficient and Interpretable Coverage Assessment

Rhitabrat Pokharel[1], **Hamid Reza Hassanzadeh**[2], **Ameeta Agrawal**[1]

[1]Department of Computer Science, Portland State University
[2]Optum AI
{pokharel, ameeta}@pdx.edu, hamid.hassanzadeh@optum.com

Abstract

Large Language Models (LLMs) have demonstrated strong capabilities in interpreting lengthy, complex legal and policy language. However, their reliability can be undermined by hallucinations and inconsistencies, particularly when analyzing subjective and nuanced documents. These challenges are especially critical in medical coverage policy review, where human experts must be able to rely on accurate information. In this paper, we present an approach designed to support human reviewers by making policy interpretation more efficient and interpretable. We introduce a methodology that pairs a coverage-aware retriever with symbolic rule-based reasoning to surface relevant policy language, organize it into explicit facts and rules, and generate auditable rationales. This hybrid system minimizes the number of LLM inferences required which reduces overall model cost. Notably, our approach achieves a 44% reduction in inference cost alongside a 4.5% improvement in F1 score, demonstrating both efficiency and effectiveness.

Introduction

Healthcare procedures encompass a wide range of diagnostic, therapeutic, and preventive services. From routine check-ups and laboratory tests to complex surgical interventions, each procedure must be accurately recorded and communicated across healthcare systems to ensure quality care and regulatory compliance. To achieve this standardization, the healthcare industry relies on codes like Current Procedural Terminology (CPT) codes. Healthcare providers, insurance companies, and other stakeholders rely on these codes to communicate effectively and process claims for a wide range of medical procedures. Accurate interpretation of CPT codes and policy documents is essential for determining whether the code aligns with specific policy provisions. Figure 1 illustrates a high level workflow involved in CPT code analysis. Given the intricate and often subjective nature of healthcare policy documents, this process can be time-consuming and susceptible to inconsistencies. These challenges highlight the need for more efficient and reliable approaches to support policy logic tracing and ensure consistency within the healthcare domain.

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

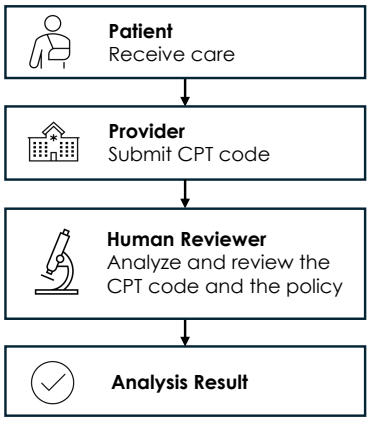

Figure 1: High level pipeline of CPT code analysis in healthcare.

Recent advances in Large Language Models (LLMs) have demonstrated considerable promise in decision-making across domains such as legal analysis and healthcare policy interpretation (Ryu et al. 2025; Pan et al. 2023; Xu et al. 2024), owing to their ability to process and interpret complex natural language. While it is generally accepted that LLMs are adept at analyzing textual information, they often face difficulties when analyzing lengthy complex texts. Effective reasoning in these contexts requires models to explicitly reference factual details and policy language, which are frequently dispersed throughout the document. Furthermore, LLMs may exhibit hallucinations and inconsistencies in their reasoning (Dahl et al. 2024).

Chain-of-thought (CoT) prompting (Wei et al. 2022) is a commonly explored technique (Kant et al. 2025) for guiding LLMs through multi-step reasoning processes. However, CoT approaches are not immune to lack of interpretability, generating inconsistent reasoning, and can be computationally expensive when applied at scale.

Expert systems have been used to simulate the reasoning abilities of human experts by encoding domain-specific knowledge and reasoning processes (Li et al. 2024; Garrido-Merchán and Puente 2025). These systems operate on structured representations of knowledge, such as facts and "if-then" rules, using deterministic logic to systematically pro-

cess knowledge. By applying predefined rules, it enables expert systems ensure consistency and interpretability.

In this paper, we introduce a neuro-symbolic approach that emulates the structured reasoning patterns of expert systems to assist human reviewers with understanding and tracing policy logic. Our method integrates neural components for processing natural language with symbolic reasoning modules that apply clearly defined rules. By translating policy terms into a machine-interpretable format, the system can organize and surface the relevant logical conditions underlying a policy. In doing so, it provides interpretable rationales that point directly back to the governing policy language. Crucially, the system does not make coverage determinations; human reviewers maintain full adjudication authority. Instead, our approach serves as a tool to find support from coverage documents to support auditable reasoning throughout the review process.

Our major contributions are as following.

- We develop a framework that supports human experts with analyzing complex documents by integrating neuro-symbolic approach.
- To support transparent tracing, we implement a coverage-aware retriever that accurately identifies and extracts governing policy language relevant to specific CPT codes.
- We provide a rule-based system that significantly lowers the cost of reasoning compared to continuous LLM inference.

Related Work

Translating Natural Language into First-Order Logic

The use of LLMs to translate natural language into formal rules has gained significant attention in recent research. Techniques such as prompting with CoT and neuro-symbolic approaches (Wei et al. 2022; Nezhad, Li, and Agrawal 2025; Nezhad and Agrawal 2025) have shown promising results in this area. Contractual language often encodes logical relationships in natural language that humans interpret with ease. Several recent systems, including CLOVER (Ryu et al. 2025), LOGIC-LM (Pan et al. 2023), LogicLLaMA (Yang et al. 2024), ProSLM (Vakharia et al. 2024), Thought Like Pro (Tan et al. 2024), Symb-CoT (Xu et al. 2024), and LLM-Tres (Toroghi et al. 2024), have explored methods for translating natural language into first-order logic representations. These approaches aim to bridge the gap between unstructured contract language and machine-interpretable rules. Our work is in line with these efforts to formalize natural language reasoning, but differs in that we employ a rule-based expert system to operationalize domain knowledge through executable reasoning rather than formal logical translation.

LLMs for Legal Reasoning

LLMs have demonstrated notable progress in supporting legal decision-making tasks, including the following examples. Guha et al. (2023) introduced LegalBench, a comprehensive suite of benchmarks to evaluate the legal reasoning capabilities of LLMs. Yao et al. (2025) proposed a reinforcement learning-based approach for legal question answering. Mishra et al. (2025) conducted an error analysis of LLM reasoning in civil procedure contexts. Dahl et al. (2024) examined the prevalence of hallucinations in LLMs applied to legal domains. Yue et al. (2024) presented LawLLM, a system offering legal reasoning services through fine-tuning techniques. Shen et al. (2025) introduced a reasoning schema for legal tasks that integrates factual grounding. Similarly, Shi et al. (2025) proposed LegalReasoner, which first identifies disputes to decompose complex cases and then performs step-wise reasoning. Our approach focuses on using actionable rules generated from text.

LLMs for Rule-Making

Cummins et al. (2025) explored the potential of using computable rules for decision-making in a specific domain, leveraging Prolog-like logical representations to encode policy logic in a machine-readable format. Their work highlights the importance of formal rule structures in enabling automated reasoning, but relies heavily on manual encoding and domain expertise for rule construction.

Building on this foundation, Kant et al. (2025) conducted an extensive evaluation of the current generation of LLMs in producing structured rules from policy documents. In their study, the authors systematically compared the decision-making capabilities of LLMs using both prompt-based approaches and rule-based frameworks. LLMs were tasked with generating structured rules from policy text, guided by human-designed schemas featuring specific attributes and helper functions. Their findings show that most LLMs exhibit markedly improved reasoning performance when integrated with explicit rule-based scaffolding, underscoring the value of structured logic in complex adjudication tasks. However, their approach is limited by the scope of the provided schemas and helper functions, constraining reasoning to facts explicitly represented in the input.

In contrast to prior approaches, our work advances this line of research by eliminating the need for human-curated schemas and helper functions. We focus on dynamic rule generation from natural language, leveraging finetuned models and symbolic reasoning to automatically extract governing policy language and generate rules with the intent to offers a scalable and cost-effective solution that reduces manual effort and reliance on frequent LLM inference.

Coverage Documents

Coverage documents serve as references that outline the criteria and provisions associated with a given policy. They are organized into multiple sections and subsections, each providing detailed information about the procedures, services, or conditions addressed by the policy. These subsections contain specific language that clarifies the scope of the policy that guides how particular provisions apply in various contexts. A sample subsection is presented in Figure 2.

Diabetes Self Management and Training/Diabetic Eye Exams/Foot Care Outpatient self-management training for the treatment of diabetes, education and medical nutrition therapy services. Services must be ordered by a Physician and provided by appropriately licensed or registered health care professionals who are authorized to prescribe such items and who demonstrate adherence to minimum standards of care for diabetes mellitus as adopted and published by the Diabetes Initiative. Benefits also include medical eye exams (dilated retinal exams) and preventive foot care for diabetes. Diabetic Self Management Items Insulin pumps and supplies and continuous glucose monitors for the management and treatment of diabetes, based upon your medical needs. An insulin pump is subject to all the conditions of coverage stated under Durable Medical Equipment (DME), Orthotics and Supplies. Benefits for blood glucose meters including continuous glucose monitors, insulin syringes with needles, blood glucose and urine test strips, ketone test strips and tablets and lancets and lancet devices are described under the Outpatient Prescription Drug Rider.

Figure 2: A sample text from *Diabetes Services* subsection of a plan document.

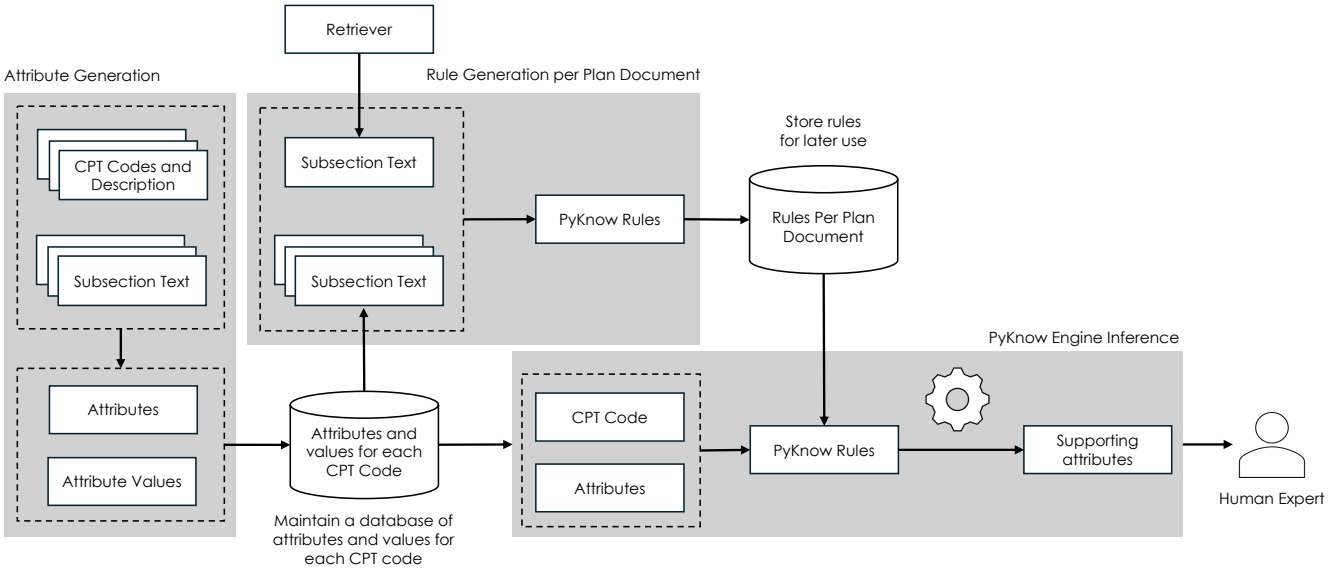

Figure 3: Pipeline of our neuro-symbolic approach. This system supports human reviewers by making the underlying policy logic interpretable.

Methodology

We define our task as follows: given a CPT code, its accompanying description, and a policy document, the objective is to generate an interpretable reasoning trace that links the code to the relevant policy language. We first describe our approach for extracting relevant coverage language from policy documents, and then discuss the process of formulating symbolic rules. Overall pipeline is shown in Figure 3.

Policy Text Retrieval Phase

A core challenge in reasoning over symbolic rules is retrieving policy language that governs coverage, rather than simply matching on topical similarity. Standard semantic search is ill-suited for this task because the signals that determine benefit status are often orthogonal to thematic content. For instance:

- A CPT for insulin pump initiation is thematically close to passages about diabetes self-management education, nutrition therapy, or endocrinology follow-ups, none of which determine its benefit status. The governing rule is more likely a short paragraph under Durable Medical Equipment or a specific policy rider.
- A CPT for continuous glucose monitoring may cluster with general diabetes monitoring advice or HbA1c screening policies, while the actual coverage clause is often a concise exclusion or a limitation found in a separate section.
- CPTs for debridement or wound care sit near content about diabetic foot care or peripheral neuropathy, whereas the decisive language is typically found in surgical necessity sections that include prior-authorization requirements.

To align retrieval with what matters for reasoning, we developed a coverage-aware retriever. Instead of optimizing for semantic likeness, we trained a cross-encoder to score subsections by their likelihood of explicitly governing a CPT's coverage, limitations, or exclusions.

Expert-Labeled Supervision. We built an internal annotation platform and engaged approximately 20 certified coding Subject Matter Experts (SMEs), with an arbitration process to ensure consistency. For each CPT across 172 Certificates

of Coverage/Summary Plan Documents (CoCs/SPDs), these SMEs selected only the passages that decide coverage. This effort produced over 1.84 million labeled (CPT, subsection, relevance) pairs, with 10% reserved for validation and the remainder ($\sim$ 1.61 million) used for training. This dataset provides the direct supervision needed to learn the nuances of policy language.

Formulation as Contrastive Multiple-Choice Ranking. The task is framed as a multiple-choice ranking problem with a contrastive objective. For a given CPT query q and its set of candidate subsections from a plan, $\{s_1, \ldots, s_n\}$, one passage s_i is labeled positive (the true coverage passage), and the rest are negatives. The model processes each query-passage pair (q, s_i) to output a logit; a softmax over all logits yields a probability distribution. Training minimizes the cross-entropy loss on the positive label:

$$\mathcal{L} = -\log p(i = \text{positive} \mid q, S)$$

This formulation, functionally equivalent to contrastive objectives like InfoNCE (Oord, Li, and Vinyals 2018), forces the model to distinguish the single governing passage from a set of highly relevant but non-dispositive distractors from the same document. Queries are constructed as:

$$\langle \text{CPT} \rangle : \langle \text{lay description} \rangle$$

and choices are the raw subsection texts.

Architecture and Training. The model is a LongformerForMultipleChoice fine-tuned from the allenai/longformer-base-4096[1] backbone. Its 1,536-token context window is sufficient to process entire subsections, including nested formatting, without truncation. Key training parameters include the AdamW optimizer (learning rate 2e-5, weight decay 0.01), bf16 mixed precision, and gradient checkpointing for memory efficiency. The model was trained for 2.5 epochs ($\sim$48 hours) on a single node with 8 x H100 GPUs, with periodic evaluation and checkpointing. To accelerate experimentation, the dataset was pre-tokenized once and reused across training runs.

Why a Cross-Encoder Is Feasible and Effective. A cross-encoder architecture, which jointly processes the query and passage through its attention layers, is essential for this task. It can capture the fine-grained interactions between a CPT code and subtle policy phrases, such as "prior authorization required," "not covered," or "limited to", which are often lost in compressed vector representations. While computationally intensive, this approach is feasible in our setting because the candidate pool per plan is small and well-defined (typically < 60 subsections across the relevant "Covered Services" and "Exclusions & Limitations" sections). Exhaustive scoring requires a trivial number of forward passes per CPT on modern GPUs, and the gains in precision are substantial. The model learns to prioritize short, auditable policy fragments that a downstream rule-based engine can deterministically parse, making the entire RAG stack coverage-aware by design.

[1]https://huggingface.co/allenai/longformer-base-4096

Inference Pipeline.

- Preparation: For each project, all subsections from the relevant plan sections are loaded as the candidate pool.
- Scoring: For each CPT, the query is constructed and scored against every candidate subsection using the trained cross-encoder. Logits are softmax-normalized across the candidate set for that CPT.
- Filtering: Passages with a probability above a threshold τ (default 0.25) are retained, capped at a maximum of five from "Covered Services" and five from "Exclusions & Limitations." In real-world deployment, if retrieval returns no passage above τ, the system should escalate the case for human review.
- Output: Results are streamed progressively, with each row containing project id, CPT code, probability, section, subsection text, and other metadata. If no passage clears the threshold for a CPT, a placeholder row is emitted to ensure downstream stages can track completeness.

Symbolic Phase

In this phase, a rule-based system is created to systematically encode the coverage criteria to help process the CPT codes. This system is implemented using PyKnow[2], a Python library for symbolic reasoning. PyKnow facilitates the creation of expert systems by offering tools to define facts (units of information), fields (data within facts), and rules (logic for reasoning over facts). In this study, we use the term "attributes" to refer to the fields. Our methodology first extracts attributes associated with each CPT code, followed by generating rules derived from coverage text, and finally executing the PyKnow inference engine to apply these rules and simulate expert reasoning over the data.

Attribute Generation. Attributes represent distinct properties that characterize a CPT code and relate it to specific policy provisions. Each attribute reflects a specific characteristic of the procedure, such as whether it pertains to mental health or preventive care. For a given coverage document, once each CPT code is mapped to its relevant subsections (in the retrieval phase), the codes are grouped by subsection. To generate attributes for each CPT code, the model is prompted to identify the properties that describe the CPT code and are likely shared with the previously grouped subsections. To facililate this, a short description of the code is also provided. Simultaneously, the model assigns default values (True or False) to these attributes based on the extracted information. Each attribute is framed as a yes/no question, meaning it should provide a clear "yes" or "no" response regarding the applicability of the attribute to the CPT code. For example, an attribute might be `is_implant`, where the value is either `True` or `False`. The prompt used for this process is detailed in Figure 4. Attributes for CPT codes are created only once, and the same attributes can be reused for new plan documents. This approach ensures scalability while minimizing costs. We use 10 plan documents during this step.

[2]https://github.com/buguroo/pyknow

You are a healthcare data architect designing a PyKnow rules engine that determines if a particular CPT code is covered under a health-insurance plan.

TASKS
1) Brainstorm every plausible Boolean attribute (i.e., a property that can be answered with True/False) that might characterize the given CPT code.

2) In addition to attributes suggested by the CPT description, also review the coverage policy terms provided, and include attributes from the policy that characterize the CPT code.

3) Use standard medical terms for the attribute names. All attribute names must follow the controlled-vocabulary pattern <is>_
Examples: is_surgical, is_outpatient
- When in doubt, default to the most widely-used medical term. Do not generate long attribute names.
- Do not create attributes whose value can be determined from the coverage plan only.
e.g. is_covered_under_dme. The attribute value should be able to be determined from the cpt description only.
- Do not skip any clinical methods mentioned in the cpt description.
- In addition, identify implicit functional context from CPT descriptions.
Example: a procedure may imply "surgery" without using the word; extract such context-aware attributes as well.

4) For each attribute, provide:
- attribute_name (snake_case)
- default_value (True or False) you would assume for the cpt code based on its CPT description

5) General instructions
- Do not use verbose attribute naming such as using "_related" or "_involved" at the end. For example,
a) Use "is_mental_health" instead of "is_mental_health_related".
b) Use "is_scopic" instead of verbose alternatives "is_scopic_procedure".
c) Use "is_prosthetic_implant" instead of "is_prosthetic_implant_involved" or "is_prosthetic_implant_included".
- Use a single, non-negated attribute when its truth value can express both presence and absence—avoid redundant negated forms.
e.g. Instead of having both is_surgical and is_non_surgical, just use is_surgical.
- In addition, create at most 1 or 2 context-aware attributes. eg. if a procedure is related to storage of reproductive item, generate is_storage and is_reproduction.
- Return ONLY the JSON list; no prose, no Markdown; no "```json".
- Do NOT include the CPT code or its description in the JSON.

INPUT
1) CPT code and its description
2) Relevant Coverage Subsection

Figure 4: The prompt used for attribute generation.

```
@Rule(Procedure(
    is_telemedicine=True,
    is_real_time=True,
    is_audio_video=True,
    is_home_based=True,
    is_non_facility=True
    )
)
def virtual_care_inclusion(self):
    self.result = "covered"
    self.category = "Virtual Care"
```

Figure 5: An example of a rule generated. Each rule is in this similar format.

Rule Creation. In this step, symbolic rules are generated using a well-structured and guided prompt based on the coverage text and relevant attributes. For each subsection, the associated CPT codes are grouped along with their attributes. These attributes help represent the coverage document in the form of PyKnow rules. The reason for incorporating attributes during rule creation is to ensure that rules are created using only the relevant attributes. Without this, attributes not associated with the CPT codes might be created, leading to potential syntax errors. For each plan document, a distinct set of rules is generated, with rules created for each subsection within the document. The prompt used during this process is in Appendix . Specific instructions are provided to ensure that the generated rules are free from syntax errors. A sample rule is presented in Figure 5.

Inference. In this step, given a CPT code and its associated attributes, we use the PyKnow engine to identify which rule is triggered. After a rule is matched, the relevant attributes are passed on to the human reviewer for further analysis.

Experimental Settings

We use internal data, which consists of coverage documents, procedure descriptions, and corresponding human-generated determinations. The coverage documents are in the form of a plain text. To ensure compliance with privacy policies, both patient information and details related to specific plan documents are fully anonymized. The eval-

Plan	Accuracy			F1 Score		
	GPT4.1	**Rule-based (ZR)**	**Rule-based (FR)**	**GPT4.1**	**Rule-based (ZR)**	**Rule-based (FR)**
Plan #1	0.88	0.87	0.81	0.93	0.93	0.90
Plan #2	0.78	0.84	0.82	0.87	0.91	0.90
Plan #3	0.77	0.79	0.89	0.86	0.88	0.94
Plan #4	0.88	0.83	0.88	0.93	0.90	0.93
Plan #5	0.83	0.90	0.92	0.90	0.94	0.96
Plan #6	0.77	0.83	0.88	0.86	0.90	0.93
Plan #7	0.88	0.88	0.90	0.93	0.93	0.94
Average	0.82	0.85	**0.87**	0.89	0.91	**0.93**

Table 1: Accuracy and F1 scores per plan. ZR = Zero-shot Retriever; FR = Finetuned Retriever. Overall, the finetuned rule-based system outperforms the other methods.

uation dataset consists of the same 814 CPT codes across 7 separate coverage documents (i.e. total of 5,698 codes) that were never used in the retriever training or attribute creation phase. The same dataset is used to get all the results.

For our baseline experiment, we use GPT-4.1 baseline with vanilla prompting along with CPT code and the entire plan document. The rule-based approach also utilizes GPT-4.1 for initial processing (attribute generation and rule-making). Additionally, we include other models such as o3 and GPT5-mini in an ablation study. For evaluation, we report accuracy and F1 scores.

Results

This section states both our results and our conclusions based on our observations of these results.

Overall Performance

To ensure that reviewers can trace how each rule is applied, we first assess the accuracy of the system's outputs and then examine the interpretability of the explanations it provides.

Table 1 shows the average accuracy and F1 scores across the seven plan documents. The rule-based method using the finetuned retriever consistently outperforms the other two approaches. Finetuning improves accuracy by an average of 2.69% and F1 score by 1.72% over the zero-shot baseline. This improvement is largely due to the finetuned retriever's alignment with the language and structure of coverage policies. Trained on a large set of expert-annotated (CPT, subsection, relevance) pairs, it improves at pinpointing the policy passages most relevant to benefit considerations. Unlike standard semantic search, which may return thematically similar but irrelevant text, the finetuned cross-encoder model prioritizes concise policy fragments that matter for symbolic reasoning. Better passage selection directly enhances the quality of both attribute generation and rule creation in the symbolic phase, since rules are built from more accurate and relevant policy language. By capturing subtle cues and specific policy limitations, the finetuned retriever provides higher-quality inputs to the symbolic engine, resulting in more precise coverage determinations.

Overall, these results demonstrate that integrating expert-labeled supervision and contrastive ranking objectives into the retriever, together with symbolic reasoning, leads to measurable gains in performance.

Interpretability. One of our goals is to support interpretability, rather than relying solely on direct inference from the model (such as interpretation made purely through prompting). For instance, as illustrated in Figure 6, consider a procedure labeled `S9212`. Under the corresponding policy, the rule-based system identifies that the `pregnancy_maternity_services` rule is relevant. The PyKnow engine highlights this rule because its conditions, `is_pregnancy=True` and `is_maternity=True`, match the attributes associated with the procedure. This traceability allows a human reviewer to see exactly which factors contributed to the decision, providing transparency and context. The system thus serves as a support tool, helping humans make informed judgments. Ultimately, the final decision remains in the hands of the human reviewer. In contrast, direct prompting does not offer this level of traceability and is more prone to hallucinations, making it less reliable for such task.

Cost Effectiveness. Our rule-based approach is designed for scalability and cost efficiency in processing large volumes of clinical codes. Attribute generation is performed once for each CPT code; with over 11,000 CPT code, and even more when extending to other code sets like HCPCS, the ability to avoid repeated model inference becomes crucial. Rule generation is required only once per coverage policy, further reducing computational demands. During inference, our system relies solely on symbolic reasoning and minimal hardware; it does not require a GPU or LLM-based inference, making it both fast and inexpensive. As shown in Table 2, LLM-based methods such as GPT-5-mini, GPT-4.1, and o3 incur substantially higher costs, with GPT-5-mini costing approximately $4,840 to process 11,000 CPT codes, compared to just $22 for the finetuned rule-based system. Even with a one-time setup cost for training (i.e. $2,680[3]), the rule-based approach remains highly cost-effective, especially as the number of codes grows. By eliminating the need for repeated LLM inference and leveraging symbolic reasoning, our method offers a robust and scalable solution

[3]Further details about the cost calculation are discussed in a later section.

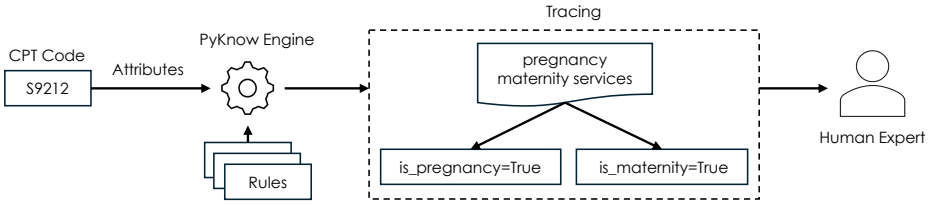

Figure 6: Overall inference workflow illustrating how our system supports human experts in identifying relevant support from coverage documents.

Model	Context Provided	Acc.	F1	Cost per 1k CPTs	Cost for 11k CPTs
GPT-5-mini (FR)	Retrieved text	0.94	0.96	$440	$4,840
GPT-4.1 (FR)	Retrieved text	0.92	0.95	$880	$9,680
O3 (FR)	Retrieved text	0.94	0.96	$880	$9,680
GPT-4.1	Entire document	0.82	0.89	$3,520	$38,720
Rule-based (ZR)	Retrieved text	0.85	0.91	$2	$22
Rule-based (FR)	Retrieved text	0.87	0.93	$2	$22

Table 2: Comparison of model accuracy, F1 score, and the inference cost for processing the CPT codes. ZR = Zero-shot Retriever; FR = Finetuned Retriever. Accuracy and F1 scores are reported on the validation set used in this study, while cost estimates reflect processing of the CPT codes. The second column indicates the type of context from the policy document supplied to each model, specifying whether the input was retrieved text passages or the full coverage document.

for large-scale coverage adjudication across diverse clinical code sets.

Performance vs Cost Tradeoff. To further investigate cost effectiveness, we evaluated the use of retrieved texts as input for direct LLM inference in Table 2. Providing only the relevant passages instead of the entire coverage document significantly reduce the number of input tokens required for each inference, thereby lowering the overall cost. This approach allowed us to assess whether our retrieval-based method remains advantageous when combined with LLMs.

The cost rates for inference and setup vary notably across the different approaches. For LLM-based methods, API pricing is based on the number of input and output tokens processed: GPT-5-mini charges $0.25 per million input tokens and $2.00 per million output tokens, while both GPT-4.1 and o3 are priced at $2.00 per million input tokens and $8.00 per million output tokens. The rule-based system that uses finetuning involves a one-time training cost of approximately $2,680, which was incurred by training on an external dataset for 48 hours using 8 × NVIDIA H100 GPUs (Azure H100 instances at 6.98/hour). Importantly, no GPUs are required for inference with the rule-based system, and the per-inference cost is extremely low, just $2.50 per 1,000 CPT codes or $22 for 11,000 codes.

LLM-based approaches such as GPT-5-mini, GPT-4.1, and o3 achieve the highest accuracy and F1 scores when using retrieved text as input, with average accuracy up to 0.94 and F1 scores up to 0.96. However, this performance comes at a substantial cost: processing 11,000 CPT codes costs $4,840 for GPT-5-mini and $9,680 for GPT-4.1 and o3. These methods also require a finetuned retriever for passage selection, adding a one-time setup cost of $2,680. In contrast, the rule-based approaches deliver competitive performance (accuracy up to 0.87 and F1 up to 0.93) at a fraction of the cost (∼$2700 for 11,000 CPT codes). Notably, models that process entire documents are both less accurate (0.82 accuracy, 0.89 F1) and dramatically more expensive ($38,720). Thus, while LLM-based methods provide slightly better accuracy with retrieved text, their inference costs scale rapidly with dataset size, making the rule-based approach far more cost-effective for large-scale adjudication tasks. Choosing between these methods involves balancing the goal of achieving optimal performance with the need to minimize operational costs. We plan to address this tradeoff in greater detail in future work.

Where do the rules fail?

Our error analysis across all procedure codes and seven plans reveals that most rule failures can be attributed to two main factors. The first and most prevalent is that the correct attribute is sometimes not incorporated into the rule creation process. This accounts for 73.5% of incorrect cases, where a rule is triggered but does not represent the correct logic. This often occurs when the attribute list is lengthy, leading the model to overlook or "forget" attributes that appear later in the input sequence, a phenomenon also observed by (Agrawal et al. 2024; Tao et al. 2025). The second failure mode, making up the remaining 26.5% of errors, involves instances where the model fails to generate a sufficient set of rules for certain samples, resulting in incomplete or missing logic and, thus, no rule being triggered for accurate coverage decisions. These findings underscore the importance of careful attribute selection and prompt engineering when constructing symbolic rules from complex policy language.

You are an expert in translating health insurance coverage documents into computable logic using the PyKnow library. Your goal is to encode the health insurance contract terms into Python rules so that we can query whether a given CPT code and description are covered under the contract.

You will be provided with either of these or one of these.
1. Terms of Inclusion and/or Exclusion:
  - A list of covered procedures/services, including any relevant coverage criteria.
  - A list of specifically excluded procedures or services.
2. Attribute list
  - A list of attributes to be used to make the rules.

TASKS:
1. Review all Terms Carefully:
  - Analyze the Terms of Inclusion, Clinical Trial Terms, and Terms of Exclusion to fully understand the scope of coverage.
  - Logical relationships, conditions, and dependencies in the coverage terms must be faithfully represented in the PyKnow rules to ensure accurate and reliable query results.
2. Translate Coverage Logic into PyKnow:
  - Define a CoverageEngine that evaluates each procedure (CPT code) and assigns exactly one of these two outcomes: covered or not covered
  - For every distinct coverage clause (e.g., each numbered or bulleted inclusion/exclusion item) create at least one explicit @Rule. Do not leave a clause unrepresented. Make rules for all the terms provided using the given attributes.
  - Aggregate related attributes with logical OR. Ensure syntax correctness while doing so. e.g. @Rule(Procedure(is_eye_related=True) | Procedure(is_visual_function=True)). You cannot use '|' inside Procedure. Do not use '&' or '~'. Here, both is_eye_related and is_visual_function serve similar purpose. This will ensure all the defined attributes are used.
  - If a procedure does not fall in any of the cases (i.e. fallback rule or the rule that is not present in the terms), do not write a rule for it, just leave it.
3. Store Results, Reasons, and Rule Names:
  - For every decision, store: The outcome under `self.result`; Category under `self.category` (this is the given subsection title)
4. Write Clean, Robust, and Complete Rules:
  - No CPT code and description pair should never match more than one rule. Handle edge cases properly.
5. General Coding Instructions:
  - Implement the coverage rules using PyKnow's @Rule decorators. No need for explanations.
  - The code should be free from syntax errors (e.g., TypeError). Avoid using self in TEST Lambdas, instead use class variable if needed.
  - If an attribute is None, this should not cause any error e.g. AttributeError.
  - No need for the main function. Avoid using Markdown formatting like ```python.
  - Enclose the code inside <code> </code> without any extra text.
  - Do not leave "class Procedure(Fact): pass" like this. You need to have attributes defined here.
6. General Instructions for Attributes:
  - Avoid documentation string and do not leave Attributes inside docstrings/comments.
  - Only use the attributes provided and do not create one on your own.
  - From the given attributes, define an attribute in the Procedure class only if it will be referenced in at least one @Rule condition.

Figure 7: The prompt used for rule generation.

Improving attribute coverage and accuracy is the most impactful path forward.

Conclusion

In this study, we introduced a framework designed to support human interpretability of coverage review by combining a coverage-aware retriever with a symbolic rule-based reasoning engine. Our findings suggest that finetuning the retriever with expert-labeled supervision and a contrastive ranking objective substantially improves its ability to surface the most relevant policy language compared to zero-shot approaches. This more accurate retrieval supports downstream steps such as attribute extraction and the identification of policy conditions that may be relevant for human reviewers. A key advantage of our approach is its efficiency: by narrowing the text that must be processed and reducing dependence on repeated LLM calls, the system provides significant cost savings while maintaining high performance. At the same time, limitations remain—for example, relevant attributes may be missed when long documents exceed input limits or when the underlying rule set does not fully capture policy nuances. Addressing these challenges will require continued refinement of context handling and rule construction. Overall, our results highlight the potential of neuro-symbolic methods to deliver scalable, interpretable, and cost-effective tools that support human reviewers by making policy logic more transparent and easier to trace.

Prompts

Figure 7 presents the structured prompts used during rule generation.

Acknowledgements

We thank Ardavan Saeedi and the anonymous reviewers for their constructive feedback.

References

Agrawal, A.; Dang, A.; Bagheri Nezhad, S.; Pokharel, R.; and Scheinberg, R. 2024. Evaluating Multilingual Long-Context Models for Retrieval and Reasoning. In Sälevä, J.; and Owodunni, A., eds., *Proceedings of the Fourth Workshop on Multilingual Representation Learning (MRL 2024)*, 216–231. Miami, Florida, USA: Association for Computational Linguistics.

Cummins, J.; Dávila, J.; Kowalski, R.; and Ovenden, D. 2025. Computable contracts for insurance: Establishing an insurance-specific controlled natural language-insurle. *Accessed: Feb*, 11: 2025.

Dahl, M.; Magesh, V.; Suzgun, M.; and Ho, D. E. 2024. Large Legal Fictions: Profiling Legal Hallucinations in Large Language Models. *Journal of Legal Analysis*, 16(1): 64–93.

Garrido-Merchán, E. C.; and Puente, C. 2025. GOFAI meets Generative AI: Development of Expert Systems by means of Large Language Models. *arXiv preprint arXiv:2507.13550*.

Guha, N.; Nyarko, J.; Ho, D. E.; Re, C.; Chilton, A.; Narayana, A.; Chohlas-Wood, A.; Peters, A.; Waldon, B.; Rockmore, D.; Zambrano, D.; Talisman, D.; Hoque, E.; Surani, F.; Fagan, F.; Sarfaty, G.; Dickinson, G. M.; Porat, H.; Hegland, J.; Wu, J.; Nudell, J.; Niklaus, J.; Nay, J. J.; Choi, J. H.; Tobia, K.; Hagan, M.; Ma, M.; Livermore, M.; Rasumov-Rahe, N.; Holzenberger, N.; Kolt, N.; Henderson, P.; Rehaag, S.; Goel, S.; Gao, S.; Williams, S.; Gandhi, S.; Zur, T.; Iyer, V.; and Li, Z. 2023. LegalBench: A Collaboratively Built Benchmark for Measuring Legal Reasoning in Large Language Models. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Kant, M.; Nabi, S.; Kant, M.; Scharrer, R.; Ma, M.; and Nabi, M. 2025. Towards robust legal reasoning: Harnessing logical llms in law. *arXiv preprint arXiv:2502.17638*.

Li, S.; Balachandran, V.; Feng, S.; Ilgen, J.; Pierson, E.; Koh, P. W. W.; and Tsvetkov, Y. 2024. Mediq: Question-asking llms and a benchmark for reliable interactive clinical reasoning. *Advances in Neural Information Processing Systems*, 37: 28858–28888.

Mishra, V.; Pathiraja, B.; Parmar, M.; Chidananda, S.; Srinivasa, J.; Liu, G.; Payani, A.; and Baral, C. 2025. Investigating the Shortcomings of LLMs in Step-by-Step Legal Reasoning. In Chiruzzo, L.; Ritter, A.; and Wang, L., eds., *Findings of the Association for Computational Linguistics: NAACL 2025*, 7795–7826. Albuquerque, New Mexico: Association for Computational Linguistics. ISBN 979-8-89176-195-7.

Nezhad, S. B.; and Agrawal, A. 2025. Enhancing Large Language Models with Neurosymbolic Reasoning for Multilingual Tasks. In H. Gilpin, L.; Giunchiglia, E.; Hitzler, P.; and van Krieken, E., eds., *Proceedings of The 19th International Conference on Neurosymbolic Learning and Reasoning*, volume 284 of *Proceedings of Machine Learning Research*, 1059–1076. PMLR.

Nezhad, S. B.; Li, Y.; and Agrawal, A. 2025. SymCode: A Neurosymbolic Approach to Mathematical Reasoning via Verifiable Code Generation. *arXiv preprint arXiv:2510.25975*.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Pan, L.; Albalak, A.; Wang, X.; and Wang, W. 2023. Logic-LM: Empowering Large Language Models with Symbolic Solvers for Faithful Logical Reasoning. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Findings of the Association for Computational Linguistics: EMNLP 2023*, 3806–3824. Singapore: Association for Computational Linguistics.

Ryu, H.; Kim, G.; Lee, H. S.; and Yang, E. 2025. Divide and Translate: Compositional First-Order Logic Translation and Verification for Complex Logical Reasoning. In *The Thirteenth International Conference on Learning Representations*.

Shen, J.; Xu, J.; Hu, H.; Lin, L.; Zheng, F.; Ma, G.; Meng, F.; Zhou, J.; and Han, W. 2025. A Law Reasoning Benchmark for LLM with Tree-Organized Structures including Factum Probandum, Evidence and Experiences. *arXiv preprint arXiv:2503.00841*.

Shi, W.; Zhu, H.; Ji, J.; Li, M.; Zhang, J.; Zhang, R.; Zhu, J.; Xu, J.; Han, S.; and Guo, Y. 2025. LegalReasoner: Step-wised Verification-Correction for Legal Judgment Reasoning. *arXiv preprint arXiv:2506.07443*.

Tan, X.; Deng, Y.; Qiu, X.; Xu, W.; Qu, C.; Chu, W.; Xu, Y.; and Qi, Y. 2024. Thought-Like-Pro: Enhancing Reasoning of Large Language Models through Self-Driven Prolog-based Chain-of-Thought. *arXiv preprint arXiv:2407.14562*.

Tao, Y.; Hiatt, A.; Seetharaman, R.; and Agrawal, A. 2025. "Lost-in-the-Later": Framework for Quantifying Contextual Grounding in Large Language Models. *arXiv preprint arXiv:2507.05424*.

Toroghi, A.; Guo, W.; Pesaranghader, A.; and Sanner, S. 2024. Verifiable, Debuggable, and Repairable Commonsense Logical Reasoning via LLM-based Theory Resolution. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y.-N., eds., *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 6634–6652. Miami, Florida, USA: Association for Computational Linguistics.

Vakharia, P.; Kufeldt, A.; Meyers, M.; Lane, I.; and Gilpin, L. H. 2024. Proslm: A prolog synergized language model for explainable domain specific knowledge based question answering. In *International Conference on Neural-Symbolic Learning and Reasoning*, 291–304. Springer.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Xu, J.; Fei, H.; Pan, L.; Liu, Q.; Lee, M.-L.; and Hsu, W. 2024. Faithful Logical Reasoning via Symbolic Chain-of-Thought. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 13326–13365. Bangkok, Thailand: Association for Computational Linguistics.

Yang, Y.; Xiong, S.; Payani, A.; Shareghi, E.; and Fekri, F. 2024. Harnessing the Power of Large Language Models for Natural Language to First-Order Logic Translation. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 6942–6959. Bangkok, Thailand: Association for Computational Linguistics.

Yao, R.; Wu, Y.; Wang, C.; Xiong, J.; Wang, F.; and Liu, X. 2025. Elevating Legal LLM Responses: Harnessing Trainable Logical Structures and Semantic Knowledge with Legal Reasoning. In Chiruzzo, L.; Ritter, A.; and Wang, L., eds., *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 5630–5642. Albuquerque, New Mexico: Association for Computational Linguistics. ISBN 979-8-89176-189-6.

Yue, S.; Liu, S.; Zhou, Y.; Shen, C.; Wang, S.; Xiao, Y.; Li, B.; Song, Y.; Shen, X.; Chen, W.; Huang, X.; and Wei, Z. 2024. LawLLM: Intelligent Legal System with Legal Reasoning and Verifiable Retrieval. In *Database Systems for Advanced Applications: 29th International Conference, DASFAA 2024, Gifu, Japan, July 2–5, 2024, Proceedings, Part V*, 304–321. Berlin, Heidelberg: Springer-Verlag. ISBN 978-981-97-5568-4.