GAVEL: AGENT MEETS CHECKLIST FOR EVALUATING LLMS ON LONG-CONTEXT LEGAL SUMMARIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) are increasingly applied in legal practice, with case summarization being a key long-context task where cases often exceed 100K tokens across multiple documents. Existing evaluation methods rely on checklist comparisons but use coarse-grained extraction that merges multiple values into single text blocks, missing partial matches when comparing them. They also overlook content beyond predefined checklist categories and lack writing style evaluation. In this paper, we introduce GAVEL-REF, a reference-based evaluation framework that improves checklist evaluation through multi-value extraction with supporting text, and further incorporates residual fact and writing-style assessments. Using GAVEL-REF, we move beyond the single aggregate scores reported in prior work to systematically evaluate 12 frontier LLMs on legal cases ranging from 32K to 512K tokens, primarily from 2025. Our detailed analysis reveals Gemini 2.5 Flash, GPT-5, and Claude Sonnet 4 achieve the best performance (around 50 $S_{\text{GAVEL-REF}}$), showing the difficulty of the task. These top models show consistent patterns: they succeed on simple checklist items (e.g., filing date) but struggle on multi-value or rare ones such as settlements and monitor reports. As LLMs keep improving and may eventually surpass human summaries, we also explore checklist extraction directly from case documents. We experiment with three different methods: end-to-end with long-context LLM, chunk-by-chunk extraction, and our newly developed autonomous agent scaffold, GAVEL-AGENT. Results show a trade-off between performance and efficiency: GPT-4.1 end-to-end performs best, while GAVEL-AGENT with Qwen3 reduces token usage by about 50%. We will release our code and annotations publicly to facilitate future research on long-context legal summarization.

1 INTRODUCTION

Large language models (LLMs) (Brown et al., 2020; Achiam et al., 2023) are now widely adopted across various industries and professions. The legal sector has been particularly active (Frankenreiter & Nyarko, 2022; Ziffer, 2023), with startups such as Harvey building AI for lawyers. Among legal applications, court document summarization stands out as both practically important and technically challenging. A single litigation case can easily involve dozens of court documents, including complaints, orders, and rulings, with a combined length exceeding 100,000 tokens, roughly equivalent to 80 news articles or a 300-page novel. Unlike news summarization, where lead sentences often suffice (Narayan et al., 2018; Liu & Lapata, 2019), or fiction books, where events can be summarized sequentially (Chang et al., 2024), legal cases require tracking interconnected arguments across multiple documents. It requires maintaining exact chronology, preserving relationships between parties, claims, and rulings, and ensuring that cross-references between filings remain accurate. Moreover, a collection of expert-written case summaries is available (Shen et al., 2022) to serve as a gold standard for this task. The combination of these factors makes legal summarization an ideal testbed for assessing LLMs' long-context capabilities; yet, it also calls for more reliable and comprehensive evaluation methodologies than those currently in use.

To evaluate summarization, researchers have moved beyond traditional n-gram metrics such as ROUGE (Lin, 2004) and BLEU (Papineni et al., 2002), developing checklist-based methods with LLM-as-judge (Min et al., 2023; Pereira et al., 2024; Lee et al., 2024; Lin et al., 2025). The most relevant recent work is ExpertLongBench (Ruan et al., 2025), which includes legal summarization

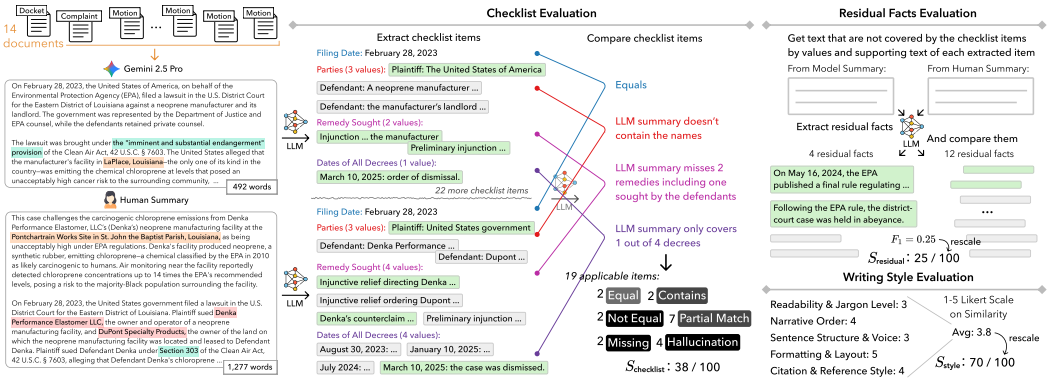

Figure 1: Example of evaluating a Gemini 2.5 Pro summary with GAVEL-REF, which contains: checklist evaluation supporting both string-wise and list-wise comparisons, residual fact evaluation, and writing-style evaluation. An interesting finding is that many modern LLMs tend to omits specific names of people or organizations—in this case, the defendant companies; and in other cases even the U.S. president's name. Light green background indicates matched values.

in its benchmark. They ask legal experts to define 26 checklist items commonly found in legal case summaries (e.g., filing date, remedy sought, decrees), and an LLM is used to extract these items from both human- and model-generated summaries for item-by-item comparison. This marks an important step toward structured and interpretable evaluation, but the approach still has two key limitations: (i) many checklist items (e.g., remedy sought) may contain multiple distinct values (see Figure 1), yet existing method treats them as a single text block, making it difficult to capture partial matches. (ii) the evaluation is restricted to predefined checklist items, overlooking additional useful content outside the checklist and other qualities such as readability or formatting. Furthermore, ExpertLongBench and other existing benchmarks (Yen et al., 2024; Ruan et al., 2025) treat legal summarization as just one task among many, reporting a single score per task. As a result, we lack detailed insights into how modern LLMs actually perform; for example, which checklist items models systematically struggle with or whether they capture non-checklist information as human experts often do. Finally, as LLMs continue to advance, they may surpass human-written summaries. This motivates deriving checklists directly from case documents to reduce reliance on human references while enabling test-time feedback. However, it is unclear from existing work whether current LLMs or agent-based methods can effectively handle this long-context extraction task.

In this paper, we address all three gaps. Firstly, we introduce GAVEL-REF (see Figure 1), which improves checklist evaluation by enabling list-wise comparison, and we further extend it with assessments of residual facts (information beyond the 26 checklist items) and writing style. We compare GAVEL-REF, with different LLMs as its backbone, against human annotators who perform the same task. Specifically, we collect 2,934 item-level annotations on 20 long summaries (averaging over 1,000 words each), 450 checklist comparison judgments, and 375 style similarity ratings, totaling 111 hours of human effort. Our results show that GAVEL-REF using open-source GPT-oss 20B (Agarwal et al., 2025) and Qwen3 (Yang et al., 2025) models achieves performance comparable to GPT-5, demonstrating that large-scale automatic evaluation can be both reliable and cost-effective.

Secondly, using GAVEL-REF, we evaluate 12 LLMs, including proprietary models (GPT-5 and Gemini 2.5) and open-source models (GPT-oss and Qwen3), on 50 cases spanning 32K to 512K tokens, far beyond the 128K limit of prior work. To reduce data contamination, 90% of cases are new from 2025 and likely unseen by the models. Our main findings are: (i) Gemini 2.5 Flash, GPT-5, and Claude Sonnet 4 achieve the best summaries with $S_{\text{GAVEL-REF}}$ score of 50 out of 100, underscoring the difficulty of long-context legal summarization. (ii) Proprietary models outperform open-source ones at the 30B scale, with open-source models such as Gemma3 (Team et al., 2025) and Qwen3 degrading more drastically as case length increases. (iii) GPT-5 best captures residual facts but tends to produce checklist-like text even when prompted for narrative style, while Claude models most closely match human style. (iv) Top models handle single-value items well, multi-value items less reliably, and struggle most with related cases and monitoring reports.

Thirdly, for extracting checklists directly from case documents, beyond standard approaches such as feeding all documents into a long-context LLM or chunking them and extracting items iteratively,

we develop a novel agent scaffold, GAVEL-AGENT. It equips LLMs with six tools for autonomously navigating documents and locating checklist items, emulating how humans process case documents. Our experiments show that end-to-end extraction with GPT-4.1 outperforms both chunk-by-chunk processing and the agent approach with Qwen3. The advantage of GAVEL-AGENT is efficiency: it uses about 50% fewer tokens than the other methods, highlighting a trade-off between performance and cost. Compared to extracting from summaries, checklist extraction from full documents still lags significantly, pointing to future work on long-context LLMs and long-horizon agents.

2 GAVEL-REF—A REFERENCE-BASED EVALUATION FRAMEWORK

We introduce GAVEL-REF (Fig. 1), an automatic, reference-based evaluation framework for legal summarization with three complementary components. First, *checklist evaluation* extracts values and supporting text for 26 items(e.g., filing date, parties, decrees). Second, *residual facts evaluation* captures and scores content beyond the checklist. Third, *writing style evaluation* compares model summaries' similarity to human references across five aspects. Prompts are in App. G.

2.1 METHOD DESCRIPTION

Checklist Evaluation. ExpertLongBench (Ruan et al., 2025) presents a checklist-based evaluation framework for long-form generation, where legal experts create a checklist of 26 key items for legal summaries. For each item c_i, an LLM extracts the corresponding information $H(c_i)$ from the model summary and $R(c_i)$ from the reference, then determines containment relationships between them. While this provides a solid foundation, we identify limitations and improve it as follows:

Improvement 1: Multi-value extraction with supporting text. We find that checklist items contain multiple values 76% of the time (e.g., several filings or factual bases in a case). However, prior method extracts all information as a single text block and performs a binary comparison. This misses partial overlaps—for example, five filings vs. five different filings with three overlaps is scored the same as a total mismatch.

To address this limitation, we restructure extraction so that each checklist item c_i yields a list of values with supporting text: $H(c_i) = \{(v_{i,1}, s_{i,1}), (v_{i,2}, s_{i,2}), \ldots, (v_{i,n}, s_{i,n})\}$, where $v_{i,j}$ is the j-th extracted value for checklist item c_i, and $s_{i,j}$ is a set of verbatim snippets grounding it. Supporting text not only justifies values but also helps us later identify residual facts that fall outside the checklist. For comparison, single-value items are judged by an LLM as equal, A contains B, B contains A, or different, while multi-value items use element-wise matching to identify overlaps and uniques.

Improvement 2: Score aggregation. When some checklist item doesn't exist in the case documents, both the model and human naturally won't include it in their summaries. However, the original method counts it as a correct match. This inflates the denominator and reduces the penalty for actual errors. As non-applicable items dilute the score calculation, errors like hallucinations or omissions of key items have less impact on the final score.

To address this issue, we compute scores based only on applicable items, defined as those present in at least one summary. The final score is: $S_{\text{checklist}} = \frac{100}{|A|} \sum_{c_i \in A} m_i$, where A is the set of applicable checklist items, and the matching score m_i is defined as:

$$m_i = \begin{cases} \begin{cases} 1 & \text{if } H(c_i) = R(c_i) \\ 0.5 & \text{if } H(c_i) \subset R(c_i) \text{ or } H(c_i) \supset R(c_i) \\ 0 & \text{otherwise} \end{cases} & \text{if single-value} \\ F_1(H(c_i), R(c_i)) & \text{if multi-value} \end{cases} \tag{1}$$

For single-value items, we assign full points for equality, half points for containment, and zero otherwise. For multi-value items, we use F_1 as the matching score.

Residual Facts Evaluation. While the checklist captures essential case information, summaries sometimes include details beyond these 26 items. To evaluate this additional content, we first identify text segments not covered by the checklist. We use two-stage matching to precisely identify uncovered text: first against the extracted values alone, then against their supporting sentences if unmatched. This prevents over-coverage—such as when a filing date's support text also contains other

legal facts. We then use an LLM to extract atomic facts (termed "residual facts") from these uncovered segments and evaluate them using the same list-wise comparison method as in our checklist evaluation. The resulting F_1 score (scaled to 0-100) is the S_{residual}.

Writing Style Evaluation. Beyond content, we measure how closely model summaries match human ones in writing style. We emphasize similarity over quality, as quality is subjective (e.g., preference for narratives vs. bullet points). Five aspects are rated on a 1–5 Likert scale (1 = completely different, 5 = identical): Readability & Jargon Level, Narrative Order, Sentence Structure & Voice, Formatting & Layout, Citation & Reference Style. We average these scores, subtract 1, and multiply by 25 to obtain S_{style} on a 0-100 scale. See Appendix C for definitions of each aspect.

2.2 The Overall Gavel-Ref Score

To combine all three components into a final score for benchmarking LLMs or use as a reward signal, we compute a weighted linear combination:

$$S_{\text{GAVEL-REF}} = (1 - r) \cdot \alpha \cdot S_{\text{checklist}} + r \cdot \alpha \cdot S_{\text{residual}} + (1 - \alpha) \cdot S_{\text{style}} \tag{2}$$

where α controls the balance between content and style, and r is the proportion of residual content in the reference summary (total residual text spans length divided by summary length). This dynamically weights $S_{\text{checklist}}$ and S_{residual} based on their relative importance in each summary—more residual content increases the weight on S_{residual}. We set α as 0.9 throughout our paper.

2.3 Meta-Evaluation of Gavel-Ref

To validate that Gavel-Ref accurately captures summary quality, we recruit four in-house annotators with legal expertise to perform the same evaluation tasks as the LLM—extracting checklist items, comparing checklist item values, and rating writing style similarity—then measure the agreement between LLM and human annotations.

Collecting Human Annotations. To evaluate LLMs' ability to *extract checklist items*, we annotate 20 long case summaries (avg. 1,093 words), as shorter ones pose a simpler challenge. Since extracting all 26 checklist items from scratch is time-consuming, annotators start from GPT-5's extractions. Using our paragraph-by-paragraph review interface modified from Thresh (Heineman et al., 2023), annotators add missing values, correct extractions and supporting text, or delete incorrect values. Each summary annotation takes approximately one hour. We collect 35 total summary annotations covering 2,934 item-level annotations. The five longest summaries (averaging 1,780 words) receive triple annotations, with adjudication by a fourth annotator. The remaining 15 summaries receive single annotations. To evaluate LLMs' ability to *compare checklist values*, annotators assess 150 item pairs from model and reference summaries (100 multi-value, 50 single-value), drawn from diverse LLMs for generalizability. For single-value pairs, they perform 4-class classification: equal, A contains B, B contains A, or different. For multi-value pairs, they match elements from list A to list B. Annotations are aggregated by majority vote: for single-value items, we take the class with $\geq$ two votes (no cases had all three labels differ); for multi-value items, we keep matches identified by $\geq$ two annotators. To evaluate LLM's ability to *rate writing style similarity*, we annotate 25 model-reference summary pairs. Annotators rate similarity across five style aspects using 1-5 Likert scales, with three annotations per pair. Final scores are the median across annotators.

All annotators are paid $18 USD per hour, with a total cost of $2K USD. Appendix D provides inter-annotator agreement results and screenshots of the annotation interfaces.

Metrics. For *checklist comparison*, we use accuracy for single-value items (4-class classification) and matching-pairs F1 for multi-value items, which measures how accurately the LLM identifies correct matches between two lists. The best comparison model is then used to evaluate *checklist extraction*, computing *S*checklist against human-extracted checklist from the same summary. We also compute word-level coverage agreement on supporting text, measuring how often model and human agree on whether words are covered by checklist items or are residual. For *writing style rating*, we report Cohen's Kappa for LLM-human agreement.

Results. We select models based on two criteria: state-of-the-art performance and open-source availability. We prioritize open-source models for cost-efficient large-scale evaluation in Section 3. We evaluate five LLMs: GPT-5 and four open-source models—Qwen3 32B, Qwen3 30B-A3B, GPT-oss 20B, and Gemma3 27B. Table 1 presents the results. GPT-5 performs best at checklist extraction with GPT-oss-20B second. Reasoning models perform better than Gemma3 27B on this task. However, Gemma3 27B outperforms all reasoning models on single string comparison and achieves comparable performance on list-wise comparison. GPT-oss 20B achieves the best alignment with human ratings of writing style. We use GPT-oss 20B for checklist extraction and style rating, and Gemma3 27B for checklist comparison in Section 3 to evaluate LLM summaries.

	Checklist Extraction		Checklist Comparison		Style
Model	$S_{\text{checklist}}$	Coverage	Single	Multi	Rating
GPT-5	**70.8**	**92.9%**	0.567	*0.847*	*0.115*
GPT-oss 20B	*65.5*	*85.9%*	0.567	0.801	**0.157**
Gemma3 27B	49.8	82.0%	**0.740**	0.841	0.091
Qwen3 32B	60.1	70.6%	0.600	0.820	0.084
Qwen3 30B-A3B	59.3	66.4%	*0.700*	**0.854**	-0.011

Table 1: Meta-evaluation results of five models in GAVEL-REF: Checklist Extraction ($S_{\text{checklist}}$ and word-level coverage agreement), Checklist Comparison (accuracy for single-value, matching F_1 for multi-value), and Writing Style Rating (Cohen's κ). **Bold**: best, *italic*: second best.

3 EVALUATION OF LLM LEGAL SUMMARIZATION WITH GAVEL-REF

Prior work (Yen et al., 2024; Ruan et al., 2025) have evaluated LLM legal summarization on legal cases up to 128K that are before 2024. As the latest LLMs now handle 1M tokens and have pre-trained knowledge up to 2025, in this work, we want to shed light on how these modern models perform on much longer context using 2025 legal cases beyond their training cutoffs. With GAVEL-REF, we evaluate 12 LLMs that span both proprietary and open-source models across 5 different case length scales: 32K, 64K, 128K, 256K, 512K tokens (measured by the GPT-4o tokenizer). For each scale, we select 10 cases whose token counts fall within $\pm 20\%$ of the target length. All cases selected are filed in 2025, except in the 512K bin where 5 cases are from before 2024 due to limited availability. Since the models have varying context limits and some cases exceed these limits, we truncate by proportionally removing tokens from the end of each document, following prior work.

3.1 BENCHMARKING RESULTS FOR 12 MODELS

Figure 2 shows GAVEL-REF evaluation results for 12 models across different case length bins.

Gemini 2.5 Flash, GPT-5 and Claude Sonnet 4 are the top three models. Proprietary models consistently outperform open-source ones by a notable margin, with GPT-oss 20B leading among open-source models. Interestingly, smaller models from the Gemini and Claude families outperform their larger siblings—Gemini 2.5 Flash beats Pro, and Claude Sonnet 4 beats Opus 4.1. This suggests that once models can handle long contexts effectively, the additional scaling on reasoning doesn't improve summarization much, unlike in reasoning-heavy tasks like coding or mathematics.

Proprietary models maintain stable checklist performance across lengths while most open-source models degrade as case length increases. All six proprietary models plus GPT-oss-20B show consistent performance regardless of length. However, Qwen3 and Gemma3 models experience significant drops on cases exceeding their native context windows.

GPT-5 performs the best on residual facts evaluation, but have diverges in writing style. GPT-5 captures more details than other models, especially on 32K-128K cases, which leads to summaries much longer than what humans write. These extra details explain why GPT-5 scores much higher S_{residual} in the 64K and 128K bins. However, it often ignores instructions to write in narrative form, instead producing section-based summaries organized by checklist items, which lowers its Formatting & Layout scores. Interestingly though, this issue fades on very long cases (256K-512K bins), where GPT-5 better follows the narrative format.

Claude models have the most human-like writing style, though all models struggle with style on very long cases. Claude Opus 4.1 leads with S_{style} of 71.7, followed by Claude Sonnet 4 at 70.7. All models perform best on 64K-128K cases for style similarity. However, on longer cases (256K-512K), every model's writing becomes less human-like, with similar drops across the board. This suggests that maintaining human-like narrative becomes increasingly difficult as case length grows.

Figure 2: Benchmarking results of 12 LLMs on long-context legal summarization with our GAVEL-REF framework across case lengths from 32K to 512K tokens. Models are ordered by $S_{\text{GAVEL-REF}}$ on all cases. Gemini 2.5 Flash leads, with all top six positions held by proprietary models.

3.2 How Top Models Handle Different Checklist Information

Figure 3 shows performance of the top five models across nine checklist groups, using the matching score m_i (Eq. 1). All models follow a similar pattern. **They are good at extracting basic case information, legal foundations, and judge details**, scoring above 0.6. This makes sense as these groups contain mostly single-value items like filing date, cause of action, type of counsel, and judge name. **Performance drops noticeably for multi-value items.** Court rulings, decrees, settlements, and factual basis (context) prove more challenging, with scores around 0.4-0.5. Models must track multiple related pieces of information scattered across lengthy documents and determine which ones are important enough to include. **The models struggle most with related cases and monitor reports,** scoring below 0.2. These items appear infrequently in the documents and often require connecting subtle references.

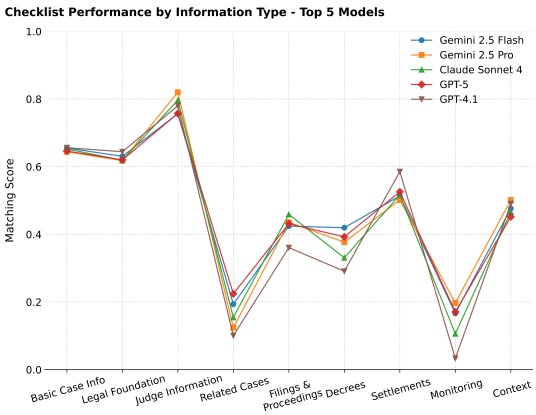

Figure 3: Top-5 LLMs' performance across checklist groups, showing struggles with multi-value items (filings, decrees, etc.) and rare items (related cases, monitor reports).

3.3 Dissecting the Top Performer: Item-Level Analysis

Figure 4 analyzes Gemini 2.5 Flash's item-level performance, showing its top and bottom 5 checklist items plus consistently over- and under-specified items (see Appendix Figure 6 for top-3 models).

Single-value items are Gemini's strength, while settlement details and monitor reports are its blind spots. Filing date leads with a near-perfect 0.99 matching score, followed by other straightforward items like monitor name (0.83) and judge name (0.76). The model's median score across all 26 items sits at 0.49. However, it struggles dramatically with settlement-related information—scoring just 0.11, 0.04, and 0.00 on various settlement items—while monitor reports score only 0.03.

Gemini 2.5 Flash tends to overspecify and underspecify checklist items with multiple values in its summaries. Important filings and trials appear in both the top-5 over&under-specified lists. This reveals that when faced with multiple values to choose from, model struggles to match human judgment about what's important. Monitor reports and settlement disputes are under-specified 100% of the time, meaning model either misses them entirely or provides less detail than humans include.

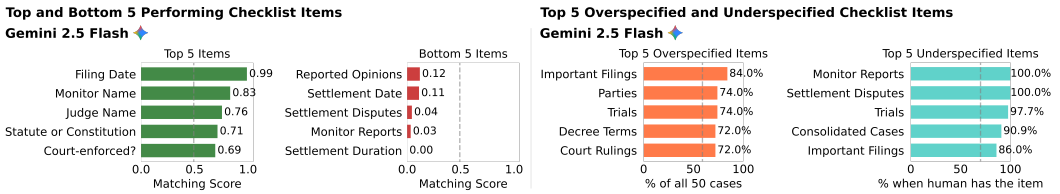

Figure 4: Gemini 2.5 Flash performance breakdown: top/bottom 5 checklist items by matching score and most frequently over/under-specified items. Overspecification measured as frequency across all 50 cases; underspecification as frequency among cases where human summary includes that item. Dashed lines are medians: 0.49 matching score, 59% overspecification, 70% underspecification.

4 Extracting Checklist from Case Documents

While reference-based evaluation effectively benchmarks summarization models, it requires hours of legal expert time per case to create human summaries, which cannot serve as a long-term gold standard once LLMs begin to surpass humans. Directly extracting checklists from case documents removes this dependency, enabling scalable evaluation, testing of superhuman models, and grounded suggestions during inference. To this end, we experiment with three methods: end-to-end extraction with long-context LLMs, processing the case documents chunk by chunk, and GAVEL-AGENT—an autonomous agent framework we develop to test whether LLMs can efficiently extract information by strategically searching and skimming rather than reading every word.

4.1 Methods

End-to-end. We concatenate all case documents in chronological order and feed them to long-context LLMs. Instead of extracting all 26 checklist items at once, we query each item individually, which gives more accurate results.

Chunk-by-chunk. We split each document into 16K-token chunks, long enough to capture most documents while fitting within modern LLM context windows (32K+). At each step, the model receives the chunk text and current checklist state, then outputs an updated state—retaining existing values or adding new ones. Like end-to-end, we process documents chronologically and extract all 26 items. This mirrors multi-agent long-context methods (Zhang et al., 2024; Zhao et al., 2024), which segment text and process chunks independently.

GAVEL-AGENT. Unlike end-to-end or chunk-by-chunk methods that make models to read everything, human experts strategically search and skim for relevant information. To mimic this, we develop GAVEL-AGENT, an agent scaffold that lets LLMs navigate documents and extract checklist items autonomously. GAVEL-AGENT provides the LLM with six tools such as read a document, run regex searches across documents, and update checklist items. At each step, the model chooses a tool or issues a stop action based on the current state and history. Standard scaffolds append each tool call and response to agent's context. While working for short tasks, this approach breaks down in long cases (256K+ tokens, 50+ calls), where the context quickly balloons and the model must track information across an increasingly unwieldy history. Instead, GAVEL-AGENT refreshes the state after each tool call, giving LLM a clean snapshot including documents explored state, recent action details, etc. GAVEL-AGENT is fully customizable: users can define any checklist items, making it easy to transfer to domains like biomedical or financial extraction.

Tools. The following are the definitions of the six tools in GAVEL-AGENT:

- `list_documents()`: Returns all available documents with their metadata such as document type and token count. It is used to provide an initial catalog of the case.
- `read_document(doc_name, start_token, end_token)`: Reads a specific token range from a document, with a maximum of 10,000 tokens per call.
- `search_document_regex(pattern, doc_name/doc_names, top_k, context_tokens)`: Searches one, multiple or all documents using regex patterns, returning the top-k matches with surrounding context (100-1000 tokens).
- `get_checklist(item/items)`: Retrieves extracted values for specified checklist items.

- `append_checklist(patch)`: Adds new values for specific checklist items, supporting multiple values per item with required evidence (verbatim text, source document, and location).
- `update_checklist(patch)`: Replaces all values for specified checklist items, used for corrections or marking items as "Not Applicable" when no relevant information exists.

Both `append_checklist` and `update_checklist` use a `patch` structure that supports batch operations. Each patch contains an array of checklist keys to update, where each key maps to an array of extracted values, and every value includes (1) the value itself and (2) an array of supporting evidence (verbatim text, source document, and location). This structure ensures traceability from extracted information back to source documents.

Context Management. At each step, the LLM is given a system prompt high-level task instruction and tool descriptions, and a user prompt that contains user instruction (e.g., "Extract all 26 checklist items"), the checklist definitions of the items to extract, a document catalog showing which parts have been explored, a summary of what has been extracted so far, and the recent action history. For action history, we maintain up to 100 tool calls: the five most recent include full responses (e.g., full text from `read_document`), while the other 95 are compressed to the tool name and brief outcome (e.g., "read 3,000 tokens", "updated filing date"). This gives the model enough awareness to avoid repeating actions while keeping the prompt compact.

4.2 Implementation Details

Model Selection. For end-to-end extraction, we use GPT-4.1 with its 1M-token context. For chunk-by-chunk extraction, we test three open-source reasoning models: GPT-oss 20B, Qwen3 32B, and Qwen3 30B-A3B. For GavelAgent, we use Qwen3 30B-A3B and GPT-oss 20B, as both support 128K+ context natively, sufficient for context management.

Gavel-Agent Configurations. It is unclear whether agents perform better extracting multiple checklist items together—potentially using each document read more efficiently—or focusing on single items for higher accuracy. To study this trade-off, we test three setups: (1) one agent extracting all 26 items; (2) 9 agents for grouped items (e.g., filing date, parties, and counsel under "Basic Case Information"); (3) 26 agents, each handling a single item. See App. B for full checklist definitions.

4.3 Meta-Evaluation

Following the evaluation of Gavel-Ref in Section 2.3, we evaluate extraction quality on 20 long cases. We use Gemma3 27B to compare each method's extracted checklist against the human-created checklist from the summary, computing the $S_{\text{checklist}}$ score. We also measure token usage (input and output) as efficiency.

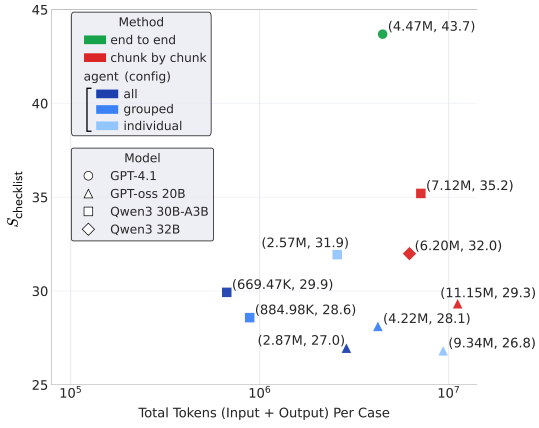

Figure 5: $S_{\text{checklist}}$ versus total token usage for different methods extracting from case documents.

Results. Figure 5 shows $S_{\text{checklist}}$ versus total token usage for each method (input and output token breakdowns are in Figure 7 in Appendix.) End-to-end with GPT-4.1 achieves the highest $S_{\text{checklist}}$ of 43.7 using 4.47M tokens. Chunk-by-chunk with Qwen3 30B-A3B ranks second with 35.2 but uses 7.12M tokens. Gavel-Agent's best configuration—individual agents with Qwen3 30B-A3B—scores 31.9 while using only 2.57M tokens, 40% fewer than end-to-end and 60% fewer than chunk-by-chunk. This performance-efficiency trade-off reflects their main difference: traditional methods read everything while agents selectively navigate documents. Across models, Qwen3 variants consistently outperform GPT-oss 20B in both chunking and agent methods. Notably, all document extraction methods fall well below the 70.8 achieved by GPT-5 extracting from human summaries in Gavel-Ref, showing significant headroom for improving both long-context models and long-horizon agents.

5 Related Work

Legal Summarization. Several datasets exist for this task. Shukla et al. (2022) release Indian and UK Supreme Court cases with human-written summaries, and Elaraby & Litman (2022) provide Canadian court opinions paired with expert summaries. Heddaya et al. (2024) collect U.S. Supreme Court opinions with their official summaries. These resources focus on single-document summarization with inputs under 16K tokens. Multi-LexSum (Shen et al., 2022) and ExpertLongBench (Ruan et al., 2025) extend this to multi-document summaries using cases from the Civil Rights Litigation Clearinghouse (CRLC), a widely used platform that offers free access to U.S. civil rights cases. Following them, we also collect cases from CRLC, focusing on 2025 filings to reduce data contamination. To better evaluate long-context capability, we construct five length ranges (32K–512K tokens) and benchmark 12 state-of-the-art LLMs with our framework GAVEL-REF, which provides fine-grained analysis of their strengths and weaknesses in long-context legal summarization.

Checklist-based Evaluation. With modern LLMs, text evaluation has moved from n-gram metrics such as BLEU (Papineni et al., 2002) or ROUGE (Lin, 2004) to LLM-based methods. One line of work (Min et al., 2023; Scirè et al., 2024) extracts atomic facts from the summaries, and verifies each fact's correctness. While precise, it is limited by inconsistent definitions of what constitutes an 'atomic' fact (Hu et al., 2024) and by poor scalability to long texts. Another line (Lee et al., 2024; Qin et al., 2024; Lin et al., 2025; Cook et al., 2024; Furuhashi et al., 2025) uses LLMs to generate task-specific rubrics and then evaluates responses against each rubric item. In domain-specific settings, human experts often design checklists that capture key information; for example, Arora et al. (2025) ask physicians to write rubrics for medical conversations. The most relevant work, ExpertLongBench (Ruan et al., 2025), introduces expert-designed checklists for 11 tasks, including 26 items for legal summarization (e.g., filing dates, court rulings). Building on this, we improve checklist extraction by requiring evidence for each item and introducing list-wise comparison. We further augment checklist evaluation with residual-fact and writing-style assessments to provide a complete picture of summary quality. Finally, we extend checklist extraction directly to case documents, reducing reliance on human summaries when evaluating future superhuman models.

LLM Agent Scaffolds. Modern LLM agents are designed as autonomous problem-solvers that plan actions and invoke tools in a multi-step loop for tasks such as web browsing (Gur et al., 2023), coding (Yang et al., 2024), or general-purpose reasoning. Several open-source scaffolds have been introduced (Xie et al., 2023; Wang et al., 2025; Lu et al., 2025; Qiu et al., 2025). For long-context processing, recent approaches segment documents into chunks or convert them into graph structures (Chen et al., 2023; Sun et al., 2024; Li et al., 2024; Zhao et al., 2024; Zhang et al., 2024), which we adopt as our chunk-by-chunk method. Inspired by how human experts read legal case documents—skimming titles, prioritizing files, and searching for keywords rather than reading everything exhaustively—we develop GAVEL-AGENT, an autonomous scaffold that equips models with six tools for navigating case documents. For context management, unlike the standard approach of continually appending tool calls and responses, we update a snapshot after each tool call and prompt the LLM with it. This design helps maintain an up-to-date state within context limits, especially when models issue 50+ tool calls in sequence, which would otherwise exhaust context quickly.

6 Conclusion

We present GAVEL-REF, a reference-based framework for evaluating long-context legal summarization that improves checklist-based evaluation with multi-value and support text extraction, and adds residual fact assessment and writing-style evaluation. In our systematic study of 12 frontier LLMs with GAVEL-REF on 2025 cases ranging from 32K to 512K tokens, we find that even the top models—Gemini 2.5 Flash, GPT-5, and Claude Sonnet 4—reach only about 50 $S_{\text{GAVEL-REF}}$, highlighting the difficulty of legal summarization. Our analysis reveals consistent patterns: models perform well on simple single-value items but struggle with multi-value and rare ones, showing key areas for improvement. To reduce reliance on human summaries, we also explore checklist extraction directly from case documents. Comparing end-to-end, chunk-by-chunk, and our proposed GAVEL-AGENT approach, we find a trade-off between performance and efficiency: end-to-end with GPT-4.1 achieves the best accuracy, while GAVEL-AGENT with Qwen3 cuts token usage by 40%-60%. Looking ahead, advancing long-context models and long-horizon agents for legal summarization and document-level extraction is key to making AI more effective in legal practice.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Rahul K Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, et al. Healthbench: Evaluating large language models towards improved human health. *arXiv preprint arXiv:2505.08775*, 2025.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Yapei Chang, Kyle Lo, Tanya Goyal, and Mohit Iyyer. BooookScore: A systematic exploration of book-length summarization in the era of LLMs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=7Ttk3RzDeu.

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. Walking down the memory maze: Beyond context limit through interactive reading. *arXiv preprint arXiv:2310.05029*, 2023.

Jonathan Cook, Tim Rocktäschel, Jakob Foerster, Dennis Aumiller, and Alex Wang. Ticking all the boxes: Generated checklists improve llm evaluation and generation. *arXiv preprint arXiv:2410.03608*, 2024.

Mohamed Elaraby and Diane Litman. ArgLegalSumm: Improving abstractive summarization of legal documents with argument mining. In Nicoletta Calzolari, Chu-Ren Huang, Hansaem Kim, James Pustejovsky, Leo Wanner, Key-Sun Choi, Pum-Mo Ryu, Hsin-Hsi Chen, Lucia Donatelli, Heng Ji, Sadao Kurohashi, Patrizia Paggio, Nianwen Xue, Seokhwan Kim, Younggyun Hahm, Zhong He, Tony Kyungil Lee, Enrico Santus, Francis Bond, and Seung-Hoon Na (eds.), *Proceedings of the 29th International Conference on Computational Linguistics*, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics. URL https://aclanthology.org/2022.coling-1.540/.

Jens Frankenreiter and Julian Nyarko. Natural language processing in legal tech. *Legal Tech and the Future of Civil Justice (David Engstrom ed.) Forthcoming*, 2022.

Momoka Furuhashi, Kouta Nakayama, Takashi Kodama, and Saku Sugawara. Are checklists really useful for automatic evaluation of generative tasks? *arXiv preprint arXiv:2508.15218*, 2025.

Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. *arXiv preprint arXiv:2307.12856*, 2023.

Mourad Heddaya, Kyle MacMillan, Anup Malani, Hongyuan Mei, and Chenhao Tan. Casesumm: a large-scale dataset for long-context summarization from us supreme court opinions. *arXiv preprint arXiv:2501.00097*, 2024.

David Heineman, Yao Dou, and Wei Xu. Thresh: A unified, customizable and deployable platform for fine-grained text evaluation. In Yansong Feng and Els Lefever (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Singapore, December 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.emnlp-demo.30/.

Qisheng Hu, Quanyu Long, and Wenya Wang. Decomposition dilemmas: Does claim decomposition boost or burden fact-checking performance? *arXiv preprint arXiv:2411.02400*, 2024.

Klaus Krippendorff. Computing krippendorff's alpha-reliability. 2011.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Yukyung Lee, Joonghoon Kim, Jaehee Kim, Hyowon Cho, and Pilsung Kang. Checkeval: Robust evaluation framework using large language model via checklist. *CoRR*, 2024.

Shilong Li, Yancheng He, Hangyu Guo, Xingyuan Bu, Ge Bai, Jie Liu, Jiaheng Liu, Xingwei Qu, Yangguang Li, Wanli Ouyang, et al. Graphreader: Building graph-based agent to enhance long-context abilities of large language models. *arXiv preprint arXiv:2406.14550*, 2024.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. Wildbench: Benchmarking LLMs with challenging tasks from real users in the wild. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=MKEHCx25xp.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*, 2019.

Pan Lu, Bowen Chen, Sheng Liu, Rahul Thapa, Joseph Boen, and James Zou. Octotools: An agentic framework with extensible tools for complex reasoning. *arXiv preprint arXiv:2502.11271*, 2025.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Singapore, December 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.emnlp-main.741/.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1797–1807, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1206. URL https://aclanthology.org/D18-1206/.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin (eds.), *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. URL https://aclanthology.org/P02-1040/.

Jayr Pereira, Andre Assumpcao, and Roberto Lotufo. Check-eval: A checklist-based approach for evaluating text quality. *arXiv preprint arXiv:2407.14467*, 2024.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. InFoBench: Evaluating instruction following ability in large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.findings-acl.772/.

Jiahao Qiu, Xuan Qi, Tongcheng Zhang, Xinzhe Juan, Jiacheng Guo, Yifu Lu, Yimin Wang, Zixin Yao, Qihan Ren, Xun Jiang, et al. Alita: Generalist agent enabling scalable agentic reasoning with minimal predefinition and maximal self-evolution. *arXiv preprint arXiv:2505.20286*, 2025.

Jie Ruan, Inderjeet Nair, Shuyang Cao, Amy Liu, Sheza Munir, Micah Pollens-Dempsey, Tiffany Chiang, Lucy Kates, Nicholas David, Sihan Chen, et al. Expertlongbench: Benchmarking language models on expert-level long-form generation tasks with structured checklists. *arXiv preprint arXiv:2506.01241*, 2025.

Alessandro Scirè, Karim Ghonim, and Roberto Navigli. FENICE: Factuality evaluation of summarization based on natural language inference and claim extraction. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.findings-acl.841/.

Zejiang Shen, Kyle Lo, Lauren Yu, Nathan Dahlberg, Margo Schlanger, and Doug Downey. Multilexsum: Real-world summaries of civil rights lawsuits at multiple granularities. *Advances in Neural Information Processing Systems*, 35:13158–13173, 2022.

Abhay Shukla, Paheli Bhattacharya, Soham Poddar, Rajdeep Mukherjee, Kripabandhu Ghosh, Pawan Goyal, and Saptarshi Ghosh. Legal case document summarization: Extractive and abstractive methods and their evaluation. *arXiv preprint arXiv:2210.07544*, 2022.

Simeng Sun, Yang Liu, Shuohang Wang, Dan Iter, Chenguang Zhu, and Mohit Iyyer. PEARL: Prompting large language models to plan and execute actions over long documents. In Yvette Graham and Matthew Purver (eds.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, St. Julian's, Malta, March 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.eacl-long.29/.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=OJd3ayDDoF.

Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luoxuan Weng, Yitao Liu, Toh Jing Hua, Junning Zhao, Qian Liu, Che Liu, et al. Openagents: An open platform for language agents in the wild. *arXiv preprint arXiv:2310.10634*, 2023.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. Helmet: How to evaluate long-context language models effectively and thoroughly. *arXiv preprint arXiv:2410.02694*, 2024.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Arik. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*, 37:132208–132237, 2024.

Jun Zhao, Can Zu, Xu Hao, Yi Lu, Wei He, Yiwen Ding, Tao Gui, Qi Zhang, and Xuanjing Huang. LONGAGENT: Achieving question answering for 128k-token-long documents through multi-agent collaboration. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Miami, Florida, USA, November 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.emnlp-main.912/.

Lee B Ziffer. The robots are coming: Ai large language models and the legal profession. In *American Bar Association*, 2023.

A Large Language Model Usage in Paper Writing

We use LLMs solely for language polishing purposes: grammar correction and paraphrasing to improve clarity and readability. We do not use LLMs to generate new content. All semantic content and scientific contributions originate entirely from the authors.

B Checklist Definitions

The followings are the definitions of the 26 checklist items used in our work, which are adapted from ExpertLongBench (Ruan et al., 2025). We group them into 9 groups.

A. Basic Case Information

1. **Filing Date**: The date when the lawsuit was first initiated with the court
2. **Parties**: Description of each plaintiff and defendant involved, including relevant positions or offices held. Use specific terms (e.g., "The city", "The parents") rather than generic terms (e.g., "The defendant", "The plaintiffs")
3. **Class Action or Individual Plaintiffs**: Whether the case involves class action plaintiffs or individual plaintiffs with descriptions
4. **Type of Counsel**: The type of legal representation (e.g., private counsel, legal services, ACLU)

B. Legal Foundation

5. **Cause of Action**: The legal basis for the lawsuit, referencing either a statute (e.g., 42 USC 1983) or a case precedent (e.g., Ex Parte Young)
6. **Statutory/Constitutional Basis**: The specific statute violated or constitutional provision allegedly violated, including the relevant clause and amendment (e.g., "Fourteenth Amendment's Equal Protection Clause" or "Commerce Clause")
7. **Remedy Sought**: The type of relief requested (e.g., declaratory judgment, injunctive relief, monetary damages)

C. Judge Information

8. **Judge Name**: First and last name of the judge

D. Related Cases

9. **Consolidated Cases**: Cases that were combined with this case for joint proceedings
10. **Related Cases**: Other cases referenced or connected to this case, listed by case code number

E. Filings and Proceedings

11. **Important Filings**: Significant motions filed, including temporary restraining orders, preliminary injunctions, motions to dismiss, and motions for summary judgment
12. **Court Rulings**: Judicial decisions on important filings such as motions to dismiss, summary judgment, preliminary injunctions, class certification, and attorneys' fees (excluding amended complaints and statements of interest)
13. **Reported Opinions**: Citations of reported opinions using shortened Bluebook format (e.g., "2020 WL 4218003"), without case name, court, or date unless from a different case
14. **Trials**: Information about trial proceedings including scheduling, outcomes, and related motions or rulings
15. **Appeals**: Whether appeals were filed, which parties appealed, to which court, and the outcomes

F. Decrees

16. **Significant Terms**: The substance of what the judge orders the defendants to do
17. **Decree Dates**: Dates when court orders or decrees were issued
18. **Duration**: How long each decree will remain in effect

G. Settlements

19. **Settlement Terms**: The substance of what the defendants agree to do in the settlement
20. **Settlement Date**: When the settlement agreement was reached

21. **Duration**: How long the settlement terms will remain in effect
22. **Court Enforcement**: Whether the court retains jurisdiction to enforce the settlement
23. **Enforcement Disputes**: Any disputes regarding compliance with settlement terms

H. Monitoring

24. **Monitor Name**: Name of any court-appointed monitor or special master
25. **Monitor Reports**: Monitor's findings regarding defendant compliance with court orders, including which terms are being met

I. Context

26. **Factual Basis**: The underlying facts and evidence supporting the legal claims, including: (i) details of relevant events (what, when, where, who), (ii) supporting evidence (physical, documentary, testimonial), and (iii) background context

C Writing Style Similarity Evaluation Details

The following are the definitions of the five aspects used in our writing style similarity evaluation. Each aspect is rated on a 1–5 Likert scale, where 5 indicates identical and 1 indicates completely different.

1. **Readability & Jargon Level**
   Compare the reading level and the balance of legal jargon vs. plain language. Consider terminology density and accessibility to non-legal readers.
   - 5 Nearly identical reading level and jargon density; same balance of technical/plain language throughout.
   - 4 Very similar complexity with minor differences in terminology or occasional variance in technical language.
   - 3 Moderate differences in accessibility; one is noticeably more technical in places but overall similar.
   - 2 Significantly different complexity; one is consistently more technical or more accessible.
   - 1 Completely different target audiences (e.g., one for legal professionals, the other for the general public).
2. **Narrative Order**
   Compare whether events are presented in the same sequence (chronological vs. thematic) and the ordering of key facts and arguments.
   - 5 Identical sequence of information; same events, facts, and arguments in the same order.
   - 4 Same overall flow with 1–2 elements reordered; core structure preserved.
   - 3 Similar general structure but several sections reordered; recognizable yet rearranged.
   - 2 Different organizational approaches with some overlap (mix of chronological and thematic).
   - 1 Completely different information architecture (e.g., one chronological, the other organized by issues).
3. **Sentence Structure & Voice**
   Compare sentence variety, active vs. passive voice, and tense consistency.
   - 5 Nearly identical sentence patterns, voice usage, and tense choices throughout.
   - 4 Very similar style with occasional differences in sentence complexity or voice.
   - 3 Moderate variation; one favors longer/shorter sentences or more active/passive constructions.
   - 2 Noticeably different styles; consistent differences in sentence variety and voice preferences.
   - 1 Completely different approaches (e.g., one varied and active; the other uniform and passive).
4. **Formatting & Layout**
   Compare use of headings, bullet/numbered lists, paragraphing, and other structural cues.
   - 5 Identical formatting choices; same use of headings, lists, and paragraph breaks.
   - 4 Very similar structure with minor variations (e.g., one extra heading or different list style).
   - 3 Similar approach but noticeable differences in execution (e.g., both use headings but at different levels/frequency).
   - 2 Different formatting philosophies; one is much more structured than the other.

1 Completely different (e.g., one heavily formatted; the other continuous prose).

5. **Citation & Reference Style**
   Compare presence, position, and formatting of case/statute citations or footnotes (inline vs. separate), citation density, and conventions.

   5 Identical citation approach; same style, frequency, and positioning.
   4 Very similar practices with minor formatting differences or occasional variation in placement.
   3 Similar philosophy but different execution (e.g., both cite cases but differ in density/positioning).
   2 Different approaches; one is substantially more reference-heavy or uses a different citation style.
   1 Completely different or incomparable (e.g., one with extensive citations, the other with none).

D Annotation Details

Inter-Annotator Agreement. For checklist extraction, the five longest summaries receive triple annotations. Agreement is measured as the average pairwise $S_{\text{checklist}}$ score across annotators, reaching 87.8 (using Gemma3 27B as the comparison model). For checklist comparison, single-value pairs achieve moderate agreement with Fleiss' $\kappa = 0.57$, while multi-value matching yields an average pairwise F1 of 0.82, indicating high consistency. For writing style similarity, Krippendorff's α (Krippendorff, 2011) across the five aspects averages 0.32.

Annotation Interfaces. Figures 8, 9, and 10 display screenshots of our human annotation interfaces for checklist extraction, checklist comparison and writing style similarity rating, respectively. The collected data are used for the meta-evaluation of GAVEL-REF and for evaluating checklist extraction from case documents methods.

E Further Analysis

Figure 6 presents the item-level performance for the top 3 models in checklist evaluation—Gemini 2.5 Flash, Pro and Claude Sonnet 4—showing their top and bottom 5 checklist items plus consistently over- and under-specified items. All three models exhibit high similar performance patterns across items.

Figure 7 presents the checklist extraction performance $S_{\text{checklist}}$ versus total, input, output token usage for each method extracting checklist from case documents.

F Implementation Details

For all language models, we use a temperature of 0.7 and top-p of 1, except for GPT-5 (where temperature cannot be changed and is fixed at 1) and Qwen3, for which we use a temperature of 0.6 and top-p of 0.95, following the official recommendations. For Gemini 2.5 Flash and Pro, we set the thinking budget to -1 (allowing the model to decide). For GPT-5, we use "high" thinking effort. For Claude Sonnet 4 and Opus 4.1, we set the thinking budget to 10,000.

We use the following versions of the proprietary models: gpt-4.1-2025-04-14, gpt-5-2025-08-07, claude-sonnet-4-20250514, claude-opus-4-1-20250805, gemini-2.5-flash (June 2025), and gemini-2.5-pro (June 2025). For open-source models, we use the instruction-tuned version of Gemma3 (Gemma3-it) and Qwen3-30B-A3B-Thinking-2507 for Qwen3 30B-A3B. Open-source models are run through vLLM Kwon et al. (2023) on 4 A40 GPUs. For all reasoning models such as Qwen3, we use the reasoning mode. Due to compute constraints, we could not run models larger than these, such as GPT-oss 120B. The total API costs is $1,500 USD.

For GAVEL-AGENT, we implement tool calls using each model's native format: ChatML for Qwen3 and Harmony for GPT-oss.

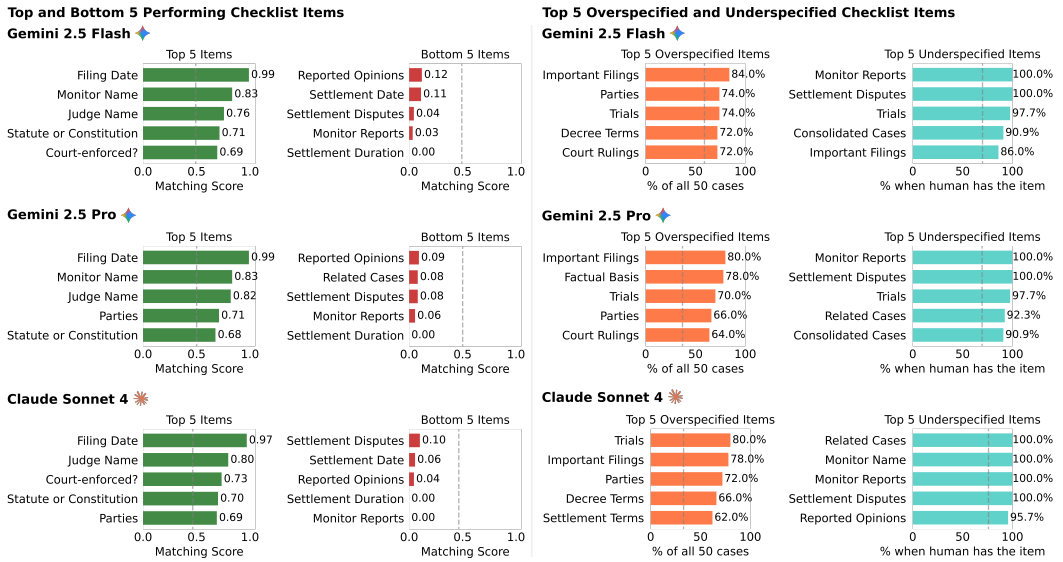

Figure 6: Performance breakdown for the top-3 models in checklist evaluation (Gemini 2.5 Flash, Gemini 2.5 Pro, and Claude Sonnet 4): top/bottom 5 checklist items by matching score and most frequently over/under-specified items. Overspecification measured as frequency across all 50 cases; underspecification as frequency among cases where human summary includes that item.

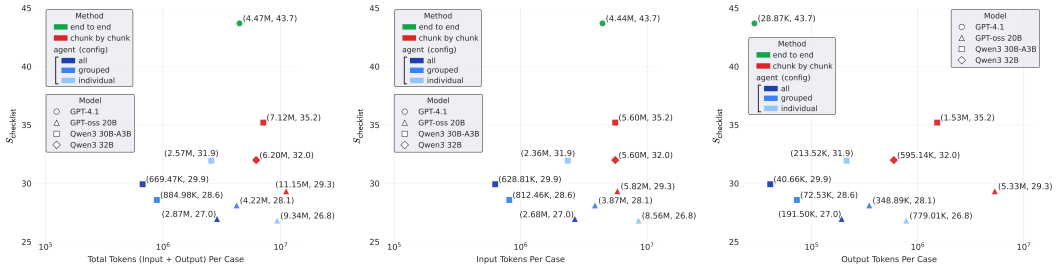

Figure 7: $S_{\text{checklist}}$ versus total token, input token, and output token usage for different methods extracting from case documents.

G PROMPTS

The following lists the prompts used in our paper.

Prompts used in GAVEL-REF. Figure 11 shows the prompt for extracting checklist items from summaries. Figures 12 and 13 show the prompts for comparing single-value and multi-value checklist items, respectively. Figure 14 shows the prompt for extracting residual facts not covered by checklist items or their supporting text. Figure 15 shows the prompt for rating writing style similarity between two summaries across five aspects.

Prompt for summarization. Figure 16 shows the prompt for legal summarization.

Prompts for checklist extraction from case documents. Figures 17 and 18 present the prompts for the end-to-end method. Figure 19 presents the prompt for the chunk-by-chunk method. Figures 20, 21, and 22 present the system prompts used in GAVEL-AGENT.

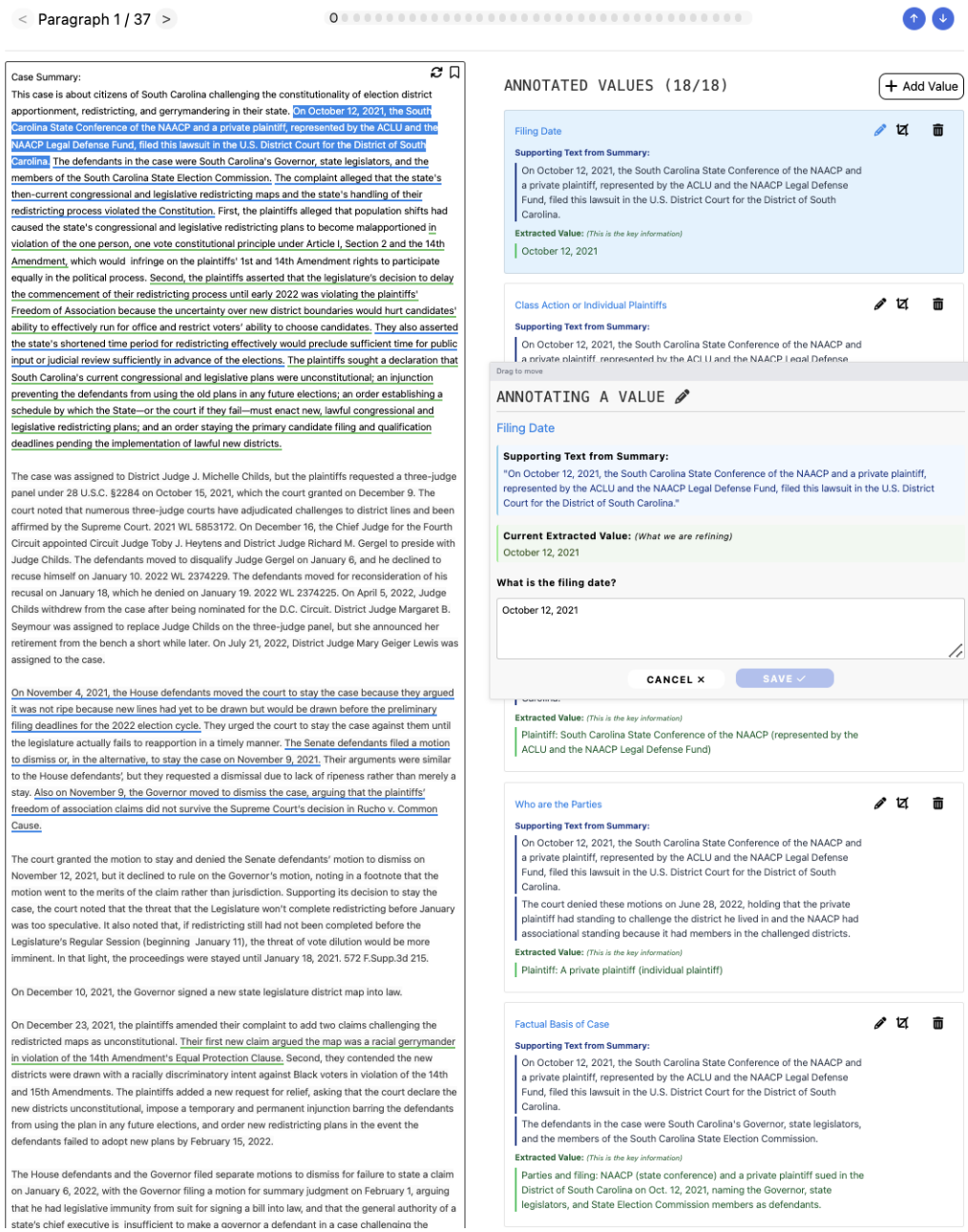

Figure 8: Screenshot of the annotation interface for checklist extraction from summaries. Annotators can add, remove, or modify checklist item values, with the process carried out paragraph by paragraph to ensure each sentence is carefully reviewed.

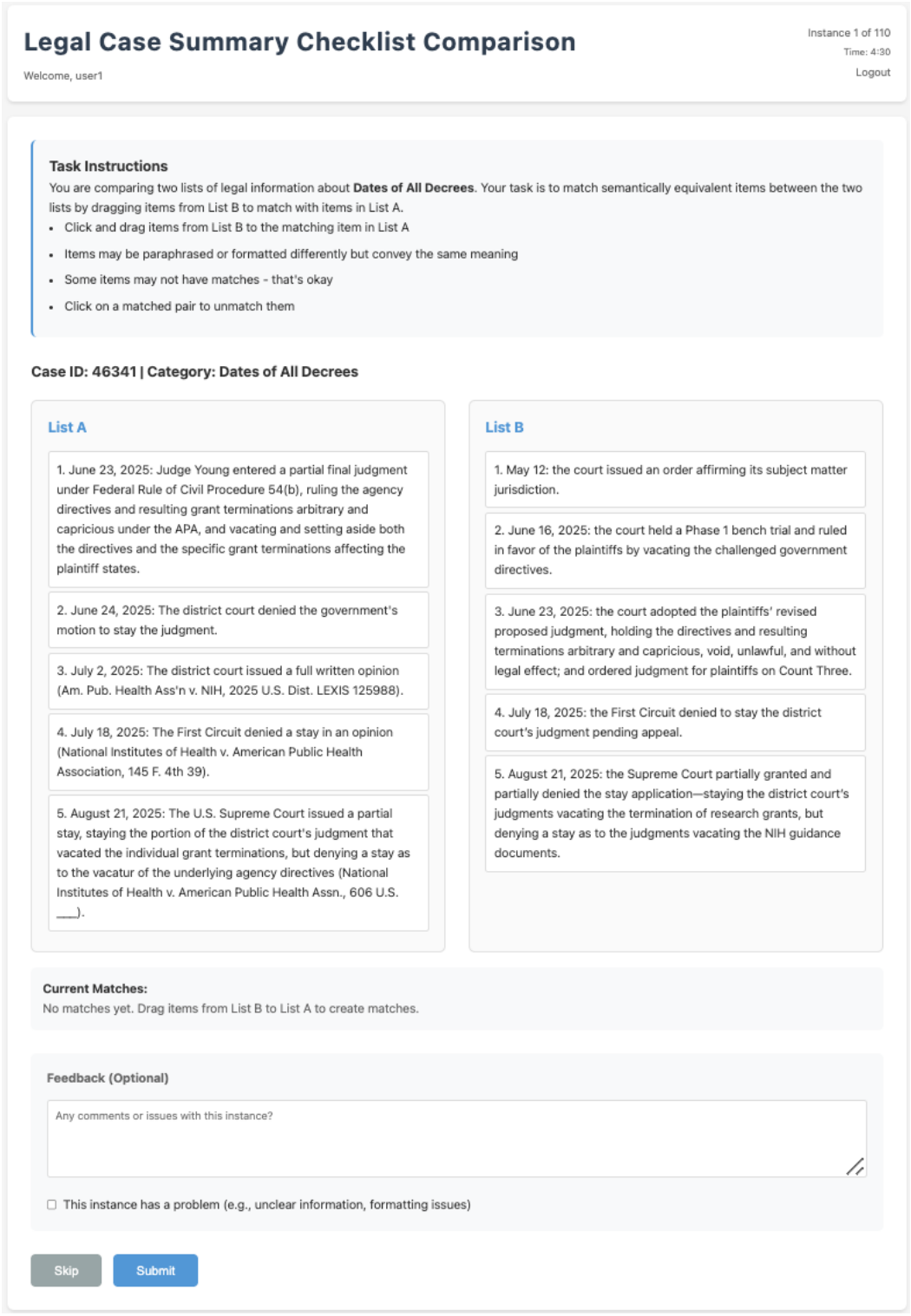

Figure 9: Screenshot of the annotation interface for checklist comparison. Annotators match items between two lists in a list-wise comparison. For string-wise comparison, where both values are strings, the middle component becomes a radio selection with four options: equal, A contains B, B contains A, or different.

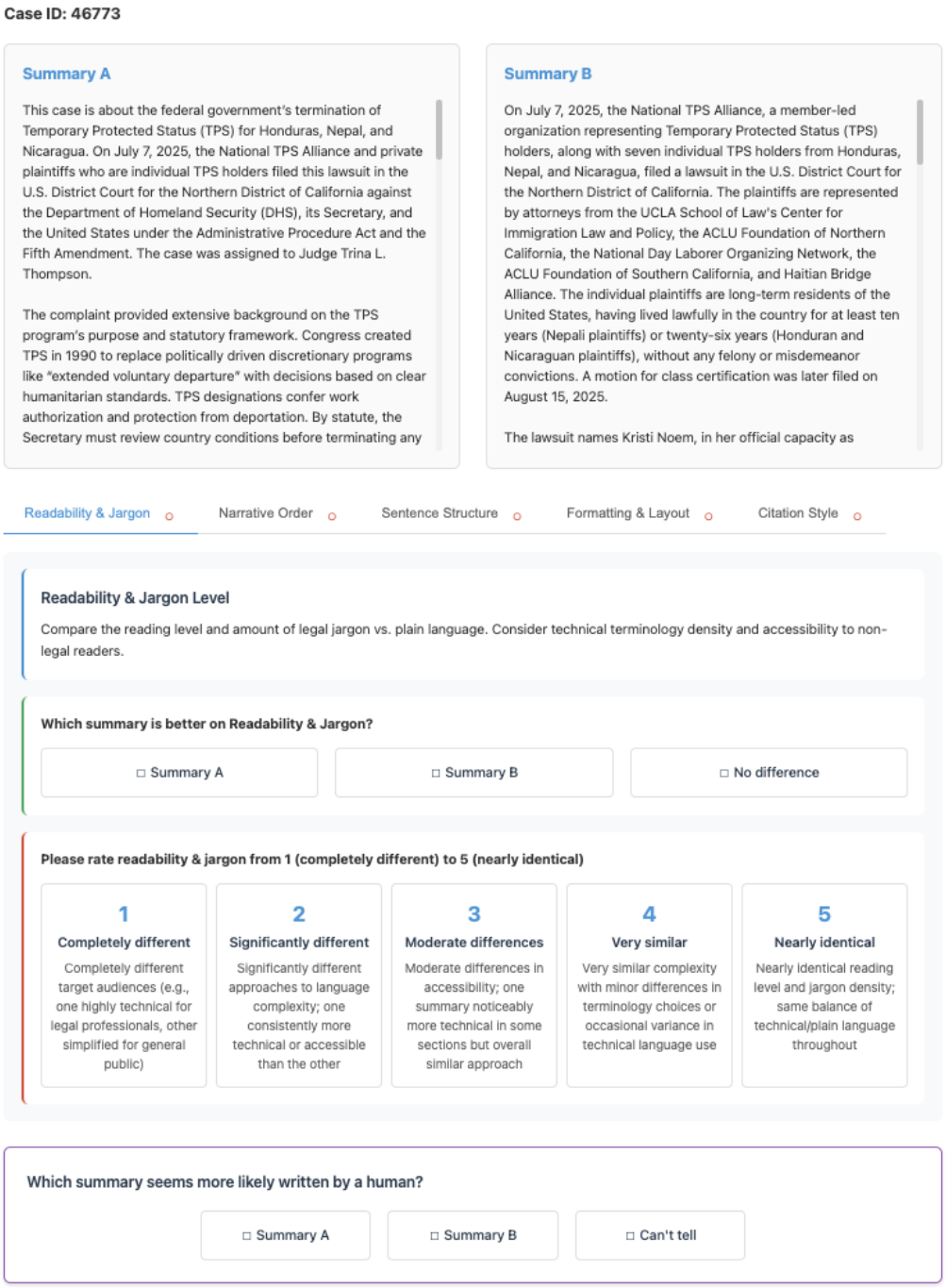

Figure 10: Screenshot of the annotation interface for rating writing style similarity. Annotators compare two summaries, providing ratings on five aspects and answering auxiliary questions such as which summary they prefer.

Prompt for Extracting Checklist from Summary

```
You are assisting a lawyer in extracting key information from a
↪  legal case summary. Given a case summary, identify
↪  {checklist_item_definition}
# Note: Do not make assumptions or add information that is not
↪  presented in the summary.

# Case Summary
{case_summary}

# Output Format
Your output should be in the following JSON format-no extra keys,
↪  no prose outside of the JSON:

```
{{
  "reasoning": "<brief analysis of the case summary and how you
  ↪ identified the relevant information or determined that none
  ↪ was present>",
  "extracted": [
    {{
      "evidence": [
        "<verbatim snippet 1>",
        "<verbatim snippet 2 (if multiple snippets are relevant)>"
        // ...
      ],
      "value": "<extracted information from the evidence>"
    }}
    // ...
  ]
}}
```
## Definitions of each part
- `reasoning`: A brief analysis of the case summary and how you
↪  identified the relevant information or determined that none was
↪  present.
- `extracted`: A list of one or more objects, each representing a
↪  distinct piece of information relevant to the checklist item
↪  (e.g., multiple court rulings, decree dates, or cited
↪  opinions). Always use a list, even if there is only one item.
- `evidence`: One or more exact text snippets copied from the case
↪  summary that support the extracted information. Always return
↪  as a list of strings.
- `value`: The extracted information.

## Rules for the JSON schema
1. **extracted** and **evidence** is always a list, even if they
↪  hold a single object.
2. Copy the **evidence** exactly as it appears in the case
↪  summary-no rewriting.
3. If the case summary contains no relevant information, output the
↪  **extracted** as an empty list:

```
{{
  "reasoning": "<brief analysis>",
  "extracted": []
}}
```
```

Figure 11

Prompt for Comparing Single-Value Checklist Item

```
You are given two pieces of legal information (A and B) about
 ↪  **{checklist_category}**, extracted from two summaries of
 ↪  the same case. Your task is to compare these pieces of
 ↪  information based on their **semantic meaning** - that
 ↪  is, what they actually convey, regardless of how they are
 ↪  worded or formatted.

# Information to Compare
## Information A:
{information_A}

## Information B:
{information_B}

# Relationship Options
Determine which of these four relationships best describes
 ↪  how A and B relate to each other:
1. **"A contains B"** - A includes all the information in B,
 ↪  plus additional information
2. **"B contains A"** - B includes all the information in A,
 ↪  plus additional information
3. **"A equals B"** - A and B convey the same information
 ↪  (semantically equivalent)
4. **"A and B are different"** - A and B contain different or
 ↪  conflicting information

# Output Format
Structure your response as follows:
**Reasoning:** Provide your detailed analysis of how the two
 ↪  pieces of information relate to each other

**Final Answer:** State one of the four options: "A contains
 ↪  B", "B contains A", "A equals B", or "A and B are
 ↪  different"
```

Figure 12

Prompt for Comparing Multi-Value Checklist Item

```
You are given two lists of legal information (A and B) about
↪  **{checklist_category}**, extracted from two summaries of the
↪  same legal case. Your task is to compare these lists based on
↪  their **semantic meaning**-that is, what each item conveys,
↪  regardless of wording, format, or phrasing.

You should identify:
1. Items that appear in **both A and B** (i.e., semantically
↪  equivalent),
2. Items that appear **only in A**,
3. Items that appear **only in B**.

# Information to Compare
## List A:
{information_A}

## List B:
{information_B}

# Output Format
Structure your response as follows:
**Reasoning:**
Provide your detailed analysis of how the two lists relate to each
↪  other. Explain any mappings between items, and how you
↪  determined whether they were equivalent or different.

**Final Answer:**
Output a valid JSON object with the following structure:

```json
{{
  "common": [
    {{"A_index": X, "B_index": Y}},
    ...
  ],
  "only_in_A": [X, ...],
  "only_in_B": [Y, ...]
}}
```

Where:
- `A_index` is the index of the item in List A,
- `B_index` is the index of the semantically equivalent item in List
↪ B,
- `only_in_A` lists the indices of items in A that do **not** appear
↪ in B,
- `only_in_B` lists the indices of items in B that do **not** appear
↪ in A.

Notes
- Both List A and B are numbered using 1-based indexing.
- Match items even if they are paraphrased or formatted
↪ differently.
- Treat legal synonyms and abbreviations as equivalent when
↪ appropriate.
- Return only valid JSON in the **Final Answer** section.
```

Figure 13

Prompt for Extract Residual Facts from Uncovered Text by the Checklist Items

```
You are assisting a lawyer in identifying key information from a
↪  legal case summary. You will be given a set of text spans
↪  extracted from the summary that may contain meaningful legal or
↪  factual content.

Your task is to extract distinct atomic facts from the given spans.
↪  Each atomic fact should be a single discrete, self-contained,
↪  and verifiable piece of information that can stand on its own.
↪  Ignore any spans that contain filler phrases, incomplete
↪  clauses, or do not convey meaningful information. If multiple
↪  spans express the same fact, extract it only once.

# Note: Do not make assumptions or add information that is not
↪  present in the spans.

# Text Spans
{text_spans}

# Output Format

Your output should be in the following JSON format-no extra keys,
↪  no prose outside of the JSON:

```
{{
  "reasoning": "<brief analysis of which spans contain meaningful
  ↪  factual information and what those facts are>",
  "extracted": [
    {{
      "fact": "<atomic fact 1>",
      "evidence_spans": [<list of 1-based span indices>]
    }},
    {{
      "fact": "<atomic fact 2>",
      "evidence_spans": [<list of 1-based span indices>]
    }}
    // ...
  ]
}}
```

Definitions of each part
* `reasoning`: A brief analysis of the spans and how you identified
↪ any meaningful atomic facts.
* `extracted`: A list of objects, each representing one atomic fact.
↪ Every object must have:
  - `fact`: A clear, concise sentence or phrase conveying a
  ↪ distinct, self-contained fact.
  - `evidence_spans`: A list of 1-based indices of the spans that
  ↪ support or directly contain the fact.

Rules for the JSON schema
{it is the same as the checklist extraction prompt.}
```

Figure 14

Prompt for Rating Writing Style Similarity on Five Aspects

```
You are given two summaries of the same legal case (Summary A and
↪  Summary B). Your task is to evaluate how similar they are in
↪  terms of structure and writing style across five specific
↪  dimensions. You should focus on **similarity** rather than
↪  quality-we want to know how alike these summaries are, not
↪  which one is better.

# Summaries to Compare
## Summary A:
{summary_A}

## Summary B:
{summary_B}

# Evaluation Dimensions with Specific Similarity Scales

{all_5_aspects_definitions}

# Output Format

Structure your response as follows:

**Analysis:**
Provide a detailed comparison for each dimension, explaining
↪  specific similarities and differences you observe between
↪  Summary A and Summary B.

**Scores:**
Output a valid JSON object with your similarity ratings:

```json
{{
  "readability_jargon": X,
  "narrative_order": X,
  "sentence_structure": X,
  "formatting_layout": X,
  "citation_style": X
}}
```

Where X is your similarity rating (1-5) for each dimension.

Important Notes
- Focus on similarity, not quality or factual correctness
- Evaluate style and structure only, ignore content differences
- Consider the summaries as a whole when rating each dimension
- Apply the scale objectively for every dimension, strictly
↪ following each definition
```

Figure 15

Prompt for Legal Summarization

```
You are given multiple documents related to a legal case. Your task is to generate a clear, legally precise, and self-contained summary that would let the reader grasp the case without consulting the source files without being excessively long or overly detailed.

Write the summary as a factual narrative. The checklist below shows what to include. Items marked "(if applicable)" should only be included when relevant. If information isn't in the documents, omit it-do not speculate.

# Legal Case Summary Checklist
{all_26_checklist_item_definitions}

# Case Documents
{case_documents}

# Output Format
Please structure your response as follows:
**Reasoning:** Briefly explain what key elements you focused on in the documents to build your summary.

**Case Summary:** A clear, legally precise narrative of the case, written in paragraph form, without being too long.

# Guidelines
* Write as a narrative in paragraph form using clear language. Use a logical order-chronological if helpful, but flexible if another sequence improves clarity.
* Include enough detail for understanding while remaining concise.
* Use accurate legal terminology but avoid jargon-write for a general audience.
* Stay strictly factual; do not add analysis beyond what appears in the record.

Now read the case documents and generate the summary following the checklist, output format, and guidelines above.
```

Figure 16

Prompt for End-to-End Extracting Checklist Item from Case Document (Part 1/2)

```
You are assisting a lawyer in extracting key information from legal
↪  case documents. You will be given multiple documents related to
↪  a legal case. Your task is to {item_description}

# Note:
- Do not make assumptions or add information that is not presented
↪  in the documents.
- When extracting evidence, quote the exact text from the
↪  documents.
- Each extracted value must be self-contained and easy to
↪  understand; include important context when available.

# Case Documents
{case_documents}

# Output Format
Your output should be in the following JSON format-no extra keys,
↪  no prose outside of the JSON:

```
{
  "reasoning": "<brief analysis of the case documents and how you
  ↪ identified the relevant information or determined that none
  ↪ was present>",
  "extracted": [
    {
      "evidence": [
        {
          "text": "<verbatim snippet 1>",
          "source_document": "<document name>",
          "location": "<page number or section>"
        },
        {
          "text": "<verbatim snippet 2 (if multiple snippets are
          ↪ relevant)>",
          "source_document": "<document name>",
          "location": "<page number or section>"
        }
        // ...
      ],
      "value": "<extracted information from the evidence>"
    }
    // ...
  ]
}
```
```

Figure 17

Prompt for End-to-End Extracting Checklist Item from Case Document (Part 2/2)

```
## Definitions of each part
- `reasoning`: A brief analysis of the case documents and how you
↪  identified the relevant information or determined that none was
↪  present.
- `extracted`: A list of one or more objects, each representing a
↪  distinct piece of information relevant to the checklist item.
↪  Always use a list, even if there is only one item.
- `evidence`: A list of evidence objects, each containing:
  - `text`: Exact text snippet copied from the case documents
  - `source_document`: The title/name of the document where this
   ↪  evidence was found
  - `location`: The page number or section identifier where the
   ↪  evidence appears
- `value`: The extracted information based on the evidence.

## Rules for the JSON schema
1. **extracted** and **evidence** are always lists, even if they
↪  hold a single object.
2. Copy the **text** in evidence objects exactly as it appears in
↪  the case documents-no rewriting or paraphrasing.
3. Always include **source_document** and **location** for each
↪  piece of evidence.
4. If the case documents contain no relevant information, output
↪  the **extracted** as an empty list:

```
{
  "reasoning": "<brief analysis>",
  "extracted": []
}
```

5. Extract information from all relevant documents-do not stop
↪ after finding information in just one document.
6. Each distinct piece of information should be a separate item in
↪ the **extracted** list.
7. If you cannot determine the specific page number or section, you
↪ may use descriptive locations like "beginning of document",
↪ "middle section", or "near the end".
```

Figure 18

Prompt for Chunk-by-Chunk Extracting Checklist Items from Case Documents

```
You are assisting a lawyer in extracting key information from legal
↪  case documents. You will be given a document chunk from a legal
↪  case. Your task is to {item_description}

# Note:
{same as the end-to-end prompt}

# Current State
This is the accumulated extraction state from previous chunks:
{current_state}

# Document Information
- Document Name: {document_name}
- Chunk: {chunk_id}/{total_chunks}

# Document Chunk
{document_chunk}

# Output Format
Your output should be in the following JSON format-no extra keys,
↪  no prose outside of the JSON:

```
{{
  "reasoning": "<brief analysis of this document chunk and how you
  ↪  identified any new relevant information or determined that
  ↪  none was present>",
  "extracted": [
    {{
      "evidence": [
        {{
          "text": "<verbatim snippet 1>",
          "source_document": "<document name>",
          "location": "Chunk {chunk_id}/{total_chunks}"
        }},
        {{
          "text": "<verbatim snippet 2 (if multiple snippets are
          ↪  relevant)>",
          "source_document": "<document name>",
          "location": "Chunk {chunk_id}/{total_chunks}"
        }}
        // ...
      ],
      "value": "<extracted information from the evidence>"
    }}
    // ...
  ]
}}
```

Definitions of each part
{same as the end-to-end prompt}

Rules for the JSON schema
{{same as the end-to-end prompt}}
```

Figure 19

System Prompt used in GAVEL-AGENT (Part 1/3)

```
You are a document extraction specialist. Your task is to extract
↪  **all checklist items specified in the snapshot** from the
↪  provided documents, citing evidence for every value.

  You operate by analyzing the snapshot and selecting **exactly ONE
  ↪  action per turn**. You must **respond with valid JSON only**
  ↪  - no prose, no extra keys.

  # Snapshot
  Provided every turn:
  - Task description
  - Checklist definitions (what items to extract; any number of
  ↪  items)
  - Document catalog with coverage statistics (and
  ↪  catalog_state/version)
  - Checklist summary (which keys are filled/empty/Not Applicable)
  - Recent action history

  # Goal
  Systematically extract all applicable checklist items with proper
  ↪  evidence.

  # Decision Policy
  Choose exactly one action each turn:
  - If the document catalog is **unknown** -> call `list_documents`.
  - If a specific document likely contains a target value, choose
  ↪  ONE:
    * `read_document` - default choice. Read a targeted window
    ↪  (<=10,000 tokens) in a document.
    * `search_document_regex` - use this when the target is clearly
    ↪  patternable (e.g., "Case No.", "Filed:", citations).
  - When you have confirmed text for one or more keys:
    - Use `append_checklist` for adds new entries for some checklist
    ↪  items.
    - Use `update_checklist` to replace the entire extracted list
    ↪  for some checklist items when you have the
    ↪  authoritative/complete set, when correcting earlier
    ↪  entries, or when setting an item to Not Applicable (see
    ↪  "Not Applicable Encoding").
  - Periodically use `get_checklist` to assess remaining gaps.
  - Stop when all keys are filled or set to Not Applicable.

  # Systematic Extraction Process
  **After each read_document or search_document_regex action:**
  - Carefully analyze the returned text to identify ALL checklist
  ↪  items that can be extracted.
  - Cross-reference the text against your checklist definitions to
  ↪  avoid missing relevant values.
  - Your next action MUST be append_checklist or update_checklist
  ↪  if you found extractable values in the text just read.

  **After each append_checklist or update_checklist action:**
  - Verify whether all extractable values from the preceding text
  ↪  were included.
  - If you notice missed values, immediately append them as the
  ↪  next action before continuing.
```

Figure 20

System Prompt used in GAVEL-AGENT (Part 2/3)

```
# Document Reading Efficiency
- **NEVER** reread fully visited documents (marked with  Fully
↪  Visited).
- **NEVER** reread token ranges already viewed (shown as "Viewed
↪  tokens: X-Y").
- When reading partially visited documents (marked with Partially
↪  Visited), read ONLY unviewed token ranges.
- Check the "Viewed tokens" list before calling read_document to
↪  avoid redundant reads.

# Write Semantics
- **Any checklist item can have multiple values**; the
↪  `extracted` field is always a list.
- **append_checklist**: add new entries; **Do not** set Not
↪  Applicable via `append_checklist`.
- **update_checklist**: replace the entire `extracted` list; use
↪  for single-valued items, complete/authoritative sets,
↪  corrections, or to set "Not Applicable".

# Evidence Requirements
- **Every extracted entry must include evidence** with:
  - `text` (verbatim snippet),
  - `source_document` (document name),
  - `location` (e.g., page, section, docket entry; include token
   ↪  offsets if available).

# Not Applicable Encoding
- Represent Not Applicable as a **single extracted entry** for
↪  that key, set **via `update_checklist`**:
  - `value`: **"Not Applicable"** (exact string; case-sensitive)
  - `evidence`: required (explicit text or a dispositive posture
   ↪  supporting Not Applicable)
- A key is treated as **Not Applicable** only if its `extracted`
↪  list contains **exactly one** entry whose `value` is "Not
↪  Applicable".
- Do **not** mark Not Applicable solely because you failed to
↪  find a value; require explicit text or logically dispositive
↪  evidence (e.g., dismissal with prejudice -> no
↪  settlement/decree; "no class certification sought" -> class
↪  action items Not Applicable).
- If later evidence shows the item **does** have real values, use
↪  `update_checklist` to replace the Not Applicable entry with
↪  the confirmed entries.

# Stop Criteria
- Stop only when every checklist key is either:
  * Complete: all relevant values present in the corpus for that
   ↪  key have been extracted, each with evidence.
  * Not Applicable: represented as a single extracted entry with
   ↪  value "Not Applicable" and supporting evidence.
- Before stopping, verify state with `get_checklist` (in a prior
↪  turn if needed) and, if consolidation is required, issue one
↪  final `update_checklist` (in a prior turn) to replace any
↪  incrementally built keys with their curated final lists. Then
↪  return the stop decision.
```

Figure 21

System Prompt used in GAVEL-AGENT (Part 3/3)

```
{{TOOL_DESCRIPTIONS}}

# Response Format
- On each assistant turn, do exactly **one** of:
  1) **Issue one function call**, or
  2) **Stop** if all applicable checklist items are fully
  ↪  extracted and any non-applicable items are marked.
- When stopping, return **only** this JSON (no extra text):
```json
{
  "decision": "stop",
  "reason": "<brief justification>"
}
```

Figure 22
```