CanLegalRAGBench: Evaluating Retrieval-Augmented Generation on Canadian Case Law

Anonymous ACL submission

Abstract

RAG-based legal assistants have been growing in popularity, but LLM hallucinations remain a key issue and potentially undermines justice. While benchmarks have been developed to evaluate progress, many rely on synthetic queries rather than realistic legal scenarios. Moreover, Canadian law remains underrepresented in existing evaluations. To address this gap, we introduce CanLegalRAGBench, a Canadian legal QA benchmark based on realistic queries and expert-annotated answers grounded in case law. Our evaluation shows that retrieval performance is sensitive to design choices and that open-source embedding models are competitive with closed source models. However, it also reveals the limitation of automatic evaluations that penalize systems for retrieving alternative relevant documents. We also find that generated answers often diverge from gold responses, either with hallucinations or by producing overly detailed or irrelevant content, with 8-29% of claims not being supported by the retrieved documents. We hope this benchmark will help drive continued progress in addressing limitations of legal RAG systems.

1 Introduction

Legal question answering is an attractive use case for retrieval-augmented generation (RAG; Lewis et al., 2020) because legal answers should be grounded in authorities that can be inspected and cited. In recent years, many RAG-based legal assistants have been developed to help laypeople and self-represented litigants understand their rights while enabling legal professionals to conduct research more efficiently. A useful system may retrieve relevant laws or case laws, identify the legal context, and answer in language appropriate to the user. However, prior work showed that LLM-based legal assistants hallucinate unsupported claims and readily accept users' incorrect assumptions (Dahl et al., 2024). At best, these errors reduce the assistant effectiveness by forcing legal professionals to spend significant time verifying the accuracy of claims. At worst, they risk distorting justice.

To measure progress on that front, the NLP community has built benchmarks for legal language understanding (Chalkidis et al., 2022; Guha et al., 2023) and case retrieval and entailment (Rabelo et al., 2022). Recent benchmarks have begun to address retrieval and end-to-end grounded generation (Pipitone and Alami, 2024; Zheng et al., 2025; Hou et al., 2025). However, most use artificial queries – such as reproducing references from citations – rather than realistic legal questions. Moreover, Canadian law is underrepresented in these evaluations. Canada's legal system has distinctive legal sources, court hierarchies, tribunal systems, and federal and provincial jurisdictional boundaries, necessitating dedicated benchmarks. Despite the availability of Canadian legal corpora, there is no public dataset designed specifically for legal RAG in Canadian law (Wallace and Rehaag, 2025; Li et al., 2024; Beauchemin et al., 2024). To bridge this gap, we propose CanLegalRAGBench, a benchmark for Canadian legal question answering with realistic user queries and expert-annotated answers grounded in Canadian case law.

We evaluated a range of RAG methods on our benchmark. Our findings show that dense retrieval and enhancements such as reranking significantly beat sparse indexing, chunk size matters significantly, and open-source embedding models are competitive with close-source models. However, standard automatic retrieval metrics are limited, and expert evaluation finds many relevant documents outside the gold set.

Generated answers differed substantially from the gold standard answers, especially when they were based off retrieved documents. Manual analysis revealed that models generated long answers that often contained irrelevant, overly-detailed claims that miss the point of the query. Finally,

Benchmark	Region	Language	Corpus	Query Type	Ground Truth Source	# Docs	# Queries
BSARD (Louis and Spanakis, 2022)	Belgium	French	law articles	Real questions (rewritten)	lawyer	22.6k	1,108
EQUALS (Chen et al., 2023)	China	Chinese	law articles	Real questions (online forums)	law student	3,081	6,914
LegalBench-RAG (Pipitone and Alami, 2024)	US	English	Varied documents	Varied: (Parsed, Automated, Real)	Automated	714	6,858
LePaRD (Mahari et al., 2024)	US	English	case law	Psuedo query	Automated	1M	4.3M
CLERC (Hou et al., 2025)	US	English	case law	Psuedo query	Automated	1.8M	105k
CANLEGALRAGBENCH (ours)	Canada	English	case law	Generated realistic queries	law student	588	532

Table 1: A summary of existing legal retrieval benchmarks.

at least 20% of the generated claims were unsupported by the documents they were referencing. Many such claims were due to reiterating the facts of the query or other connecting ideas, though there were various categories of hallucination such as misapplication of law and misunderstanding of scope and authority. We hope this benchmark will support future work in addressing remaining gaps in legal RAG systems.[1]

2 Background

Many existing legal benchmarks focus on narrow legal contexts – such as merger agreements in the US (Wang et al., 2023); but there are also evaluation suites such as LexGLUE (Chalkidis et al., 2022), LegalBench (Guha et al., 2023), and LawBench (Fei et al., 2024) that test various legal contexts and capabilities such as legal classification, extraction, NLI, and QA. Classification tasks most commonly involve legal judgment prediction (Niklaus et al., 2021; Masala et al., 2021; Semo et al., 2022). Information extraction tasks include named entity recognition and terminology extraction (Pais et al., 2021; Pham et al., 2021; Kalamkar et al., 2022), and extraction of structured information from documents (Simonson, 2021; Wenger et al., 2021; Hendrycks et al., 2021). As models' reasoning abilities improved, benchmarks were developed to measure legal reasoning, for instance identifying legal violations (Hagag et al., 2024).

More recent work has shifted toward legal question answering. BarExamQA (Zheng et al., 2025) consists of multiple-choice US bar exam questions with supporting gold passages from case laws, online resources, and textbooks. Of particular relevance to our work are benchmarks that evaluate retrieval and answer generation. We situate our benchmark against those in Table 1. Notably, few benchmarks focus on legal case retrieval (Mahari et al., 2024; Hou et al., 2025), which is a crucial

[1]We will release the benchmark, document collection, automated generation and evaluation pipeline code, and the expert annotations upon publication.

Statistic	Value
Total queries	532
Total ground truth (query, document) pairs	3,193
Total unique ground truth documents	588
Jurisdictions:	
Ontario	28%
British Columbia	22%
Alberta	22%
Other	28%
Mean ground truth docs per query	6.02
Std. dev. ground truth docs per query	4.47
Min. ground truth docs per query	1
Median ground truth docs per query	4
Max. ground truth docs per query	23

Table 2: Data statistics of CANLEGALRAGBENCH.

component of legal reasoning in common law jurisdictions, where outcomes rely on precedent.[2]

Those same two benchmarks constructed pseudo queries based on citations in the source document. This designs support creating much larger-scale datasets, but the resulting queries don't represent the type of questions that users pose to legal AI systems. Natural-language legal queries (Ravichander et al., 2019; Louis and Spanakis, 2022; Chen et al., 2023) are therefore important for practical evaluation. Our benchmark addresses this gap by combining natural queries with Canadian case law retrieval documents, enabling more realistic evaluation of legal retrieval-augmented generation.

3 CANLEGALRAGBENCH

We introduce CANLEGALRAGBENCH, a benchmark comprising of legal queries grounded in Canadian case law. The goal of CANLEGALRAGBENCH is to evaluate RAG-based systems on realistic queries that users of a legal research tools would ask. To create the data, we employed a 6-step process, illustrated in Fig. 1 and detailed below. First, we sample a random case law and a user persona, and condition on them to generate a seed query. Next, we generated variations of the query to test models' sensitivity to point of view (e.g.,

[2]ECtHR-PCR (T.y.s.s. et al., 2024) is a dataset for case retrieval, but without generation.

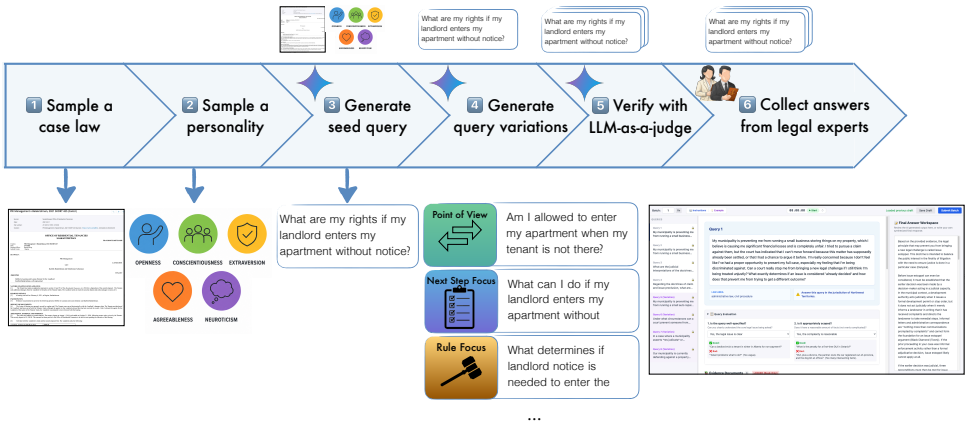

Figure 1: The pipeline used for creating CANLEGALRAGBENCH.

layperson vs. lawyer) and small but meaningful factual changes. We used an LLM-as-a-judge to filter out low quality queries. Finally, we collected the answers to the queries from legal experts. We report data statistics in Table 2.

① Sample a case law. Each seed query in our benchmark is based on an existing case. We use the A2AJ Open Canadian Legal Data (CLD; Wallace and Rehaag, 2025), a publicly available collection of 200k+ Canadian court decisions. To remove noisy documents and keep the computational requirements reasonable, we filtered out any documents in the top or bottom quartile in terms of length, keeping only the middle 50%. We selected a subset of the courts and tribunals that represent common legal disputes encountered by the general public (BCSC, SST, CHRT, TCC), and sampled documents equally from each.

② Sample a user persona. User personas were sampled using low, medium, and high amounts of each personality from the Big Five personality traits (McCrae and John, 1992). Different prompts given to the query generator were used to vary this. For all query generations we use DSPy (Khattab et al., 2023) to template our prompts.

③ Generate seed query. Given a case law and a user persona, we prompt `Gemini 2.5 Flash` to simulate a legal query from this user that can be answered by the given case law. See Prompt B.1 in the appendix. To get provincial coverage, we sampled from all provinces and territories except Quebec which follows civil law (see Appendix C.2 for the composition). To add provinces to queries, we simply prepend "I am in {province}."

④ Generate query variations. We prompted `Gemini 2.5 Flash` to generate 8 variations of each seed query. We generate four variations where the facts remain the same – thus a robust system is expected to retrieve the same case laws – but where the focus of the question or the user persona change. For example, the seed query "What are my rights if my landlord enters my apartment without notice?" in Fig. 1 can also be asked from the point of view of a landlord checking if they are allowed to enter their apartment when the tenant is not there. The other four variations change a small fact in the seed query, which may require systems to retrieve different case laws. Finally, within each batch of 8 queries, half of the queries simulate a legal professional and half simulate a lay person. See the appendix for prompts B.2 and B.3, and query variations (Appendix C.3).

⑤ Verify with LLM-as-a-judge. We used `Gemini 2.5 Pro` to filter out ∼20% of queries that are self-contradictory, nonsensical, overly complex, hinting towards the document with references, or otherwise unfit for a retrieval setup. See Prompts B.4 and B.5 in the appendix.

⑥ Collect answers from legal experts. We hired 3 final year law students from Canadian universities and a Canadian paralegal from Upwork to provide the ground truth answers for our benchmark. Annotators took 2-3 hours to complete a batch of 8 queries, and were paid $20 CAD per hour. Due to the high cost of expert annotation, each example was annotated by a single annotator. To ensure the quality of annotations, we calculated n-gram overlap between queries, answers, and snippets and manually reviewed and filtered around 30 flagged examples with obvious errors, such as answers that fail to target the query. Our evaluation

in Sec. 4 consists of both automatic evaluation that treats the expert annotation as reliable ground truth, and expert evaluation that explores the extent that experts miss relevant information.

4 Retrieval

We present the document collenction (§4.1), the baselines (§4.2), evaluation metrics (§4.3), and results (§4.4) for the retrieval task, and analyze the performance and errors (§4.5).

4.1 Document Collection and Chunking

To facilitate testing RAG systems on our benchmark, CANLEGALRAGBENCH includes additional distractor case laws, randomly selected in equal proportions from CLD and a second dataset

Since at the time of construction, CLD only comprised 11 sources (generally covering BC, Ontario, federal, and refugee law), we supplemented it with a second dataset of public court decisions across Canadian jurisdictions, which was provided to the authors by a private company.[3] This dataset covers additional jurisdictions and more diverse legal issues. We applied various strategies to remove duplicate case laws, including citation normalization, case normalization, stripping whitespaces, and manually comparing documents from the same year and similar lengths. All chunking is implemented as follows. We test various values of max character sizes (1024–8192). We chunk documents via recursive text splitting, always with 128 character overlap. The text splitter uses hierarchical text breaks such as newlines, periods and spaces.

4.2 Baselines and Evaluated Models

We benchmark various retrieval systems on CANLEGALRAGBENCH, including sparse retrieval, dense retrieval, and enhancements such as reranking and IterRetGen, which are detailed below. Table 14 lists all the models we evaluated.

Sparse Retrieval. We use BM25, an established keyword-based retrieval algorithm which is still considered a strong baseline (Robertson et al., 1994). Since legal documents are generally long, we use the BM25L variant which is specifically designed for long documents (Lv and Zhai, 2011).

Dense Retrieval. Dense retrieval works by embedding document chunks and retrieving based on vector similarity (Lewis et al., 2020). Our choice of embedding models is based on high ranking on the legal embedding benchmark (Butler et al., 2025), and includes generally strong open source models (`Qwen 3 Embedding 8B` (Zhang et al., 2025) and `EmbeddingGemma` (Vera et al., 2025)); closed models (`Gemini Embedding 001`, `Gemini Embedding 2` (Zhang et al., 2025)); and closed legal model (`Kanon 2` (Butler et al., 2025)).

Reranking. Reranking is a common way to improve retrieval methods. A typical reranker takes the query and retrieved documents and processes them together using a cross-encoder transformer, to output a relevance score for each pair. We use `Kanon 2 Reranker`, which is proprietary. We implement reranking by retrieving $4k$ documents, reranking, and retaining the top k for evaluation.

Hybrid Retrieval. We retrieve the top k documents for each of the sparse and dense retrievers, deduplicate them, rank using reciprical rank fusion (Cormack et al., 2009) or the `Kanon-2` reranker, and retain the top k documents.

Iterative Retrieval. Iter-RetGen (Shao et al., 2023) is a retrieval enhancement that iteratively generates an answer and retrieves relevant documents using the query-answer concatenation as the new query. The generated answer should share more similarity with documents than queries and improve retrieval. We do all runs at 3 iterations, and use `Gemini Flash-2.5` for answer generation (see Prompt B.6 in the appendix). For brevity we will abbreviate as IRG.

4.3 Evaluation Metrics

Automatic Evaluation. We report recall@k, MRR (MRR; Radev et al., 2002), and nDCG (nDCG; Järvelin and Kekäläinen, 2002). These metrics measure the ability to find correct documents and rank them appropriately. Following prior work (Louis and Spanakis, 2022; Hou et al., 2025), we consider recall and nDCG as the primary metrics (nDCG is more nuanced than MRR as it looks at the ranking of all top-k retrieved documents; and recall is complementary to all other metrics). We report all metrics for the main experiments, and perform analysis based on nDCG and recall, with $k = 10$. All aggregated metrics are macro averages as calculated in Eq. 1:

$$\text{Score} = \frac{1}{|Q|} \sum_{q \in Q} M(q, D_r^q, D_{GT}^q) \qquad (1)$$

[3]The portion of the data used in this work will be made available upon publication.

LLM	Chunk Size	Method	Rerank	MRR	Recall@10	Recall@25	nDCG@10	nDCG@25
–	Doc	Sparse (BM25)	✗	0.245	0.193	0.309	0.163	0.208
Qwen	8192	IterRetGen	✓	<u>0.632</u>	0.406	0.601	**0.474**	<u>0.527</u>
Gemma	8192	IterRetGen	✓	0.613	<u>0.431</u>	0.595	<u>0.471</u>	0.510
Gemini2	4096	IterRetGen	✗	**0.661**	**0.456**	**0.631**	0.521	**0.560**
Kanon2	8192	Dense	✓	0.590	0.406	<u>0.624</u>	0.449	0.514
Qwen	1024	Hybrid	✓	0.563	0.364	0.548	0.426	0.477

Table 3: Best performing configurations for each embedding model family on CANLEGALRAGBENCH. Full retrieval results are available in Table 10 in the appendix. **Bold** = overall best, <u>underlined</u> = 2nd.

where Q is the set of queries in CANLEGALRAGBENCH, M is the metric, and D_r^q and D_{GT}^q are the retrieved and gold standard documents for each query. We map retrieved chunks to their documents and report metrics at the document level.

Expert Evaluation. While automatic evaluation provides a general comparison of baselines, it is possible that automatic evaluation unnecessarily penalizes retrieving relevant documents not in the gold set. To investigate this possibility, we perform manual evaluation. We made a stratified sample of 30 queries by province first and by legal issue second using Gemini-2.5-Pro for categorization. We have our expert annotators evaluate the retrieved documents of 3 best performing configurations according to the automatic evaluation (`Qwen 8192 chunk size IRG + Rerank`, `Gemini-Embedding-2 4096 chunk size IRG`, and `Kanon2 8192 chunk size + rerank`). We select a configurations of different embedding models for diverse evaluation. We instructed the expert annotators (§3) to judge retrieved documents for relevance to the query. Documents judged as relevant are added to the gold standard documents, and we report recalculated metrics after expert evaluation for this subset.

4.4 Results

We report the overall retrieval performance on CANLEGALRAGBENCH, and look into the effect of retrieval method and embedding model on the performance.

General Trends. We tested 46 combinations of method, embedding model, and chunk size. The complete results of all evaluated configurations are reported in Table 10 in Appendix D.1. Here, we report the best performing configuration for each embedding model family in Table 3, based on the average across evaluation metrics.

The best dense retrieval based methods dramatically outperform sparse retrieval, and the hybrid approach seems to also fall behind dense retrieval. Enhancements such as IterRetGen and reranking improve upon a vanilla dense retrieval approach but inconsistently. **Large chunk sizes generally perform better** (8192 performs very well) – suggesting a good trade off of preserving legal context while adding noise – but it is also model and method dependent. Finally, with respect to model family, **`Gemini-2` performs best across most evaluation metrics**, with `Gemma` and `Qwen` closely following, suggesting that open-source embedding models are close in quality to closed source models. Interestingly, the `Kanon-2` embeddings, which is specialized for the legal domain, slightly underperforms other embedding models.

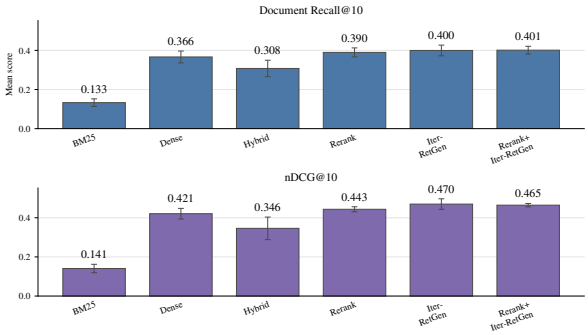

Figure 2: Average recall@10 and nDCG@10 for each retrieval method, aggregated across models and chunk sizes, with standard deviation error bars.

Best retrieval method. We report the performance of each retrieval method averaged across embedding model and chunk size in Fig. 2. Reranking, IterRetGen, and their combination, perform highly. Vanilla dense retrieval is marginally behind. Sparse and hybrid retrieval underperform. The hybrid method performs worse than dense retrieval on both metrics due to poor BM25 performance.

Table 10 shows that the effect of reranking on performance is configuration-dependent. It consis-

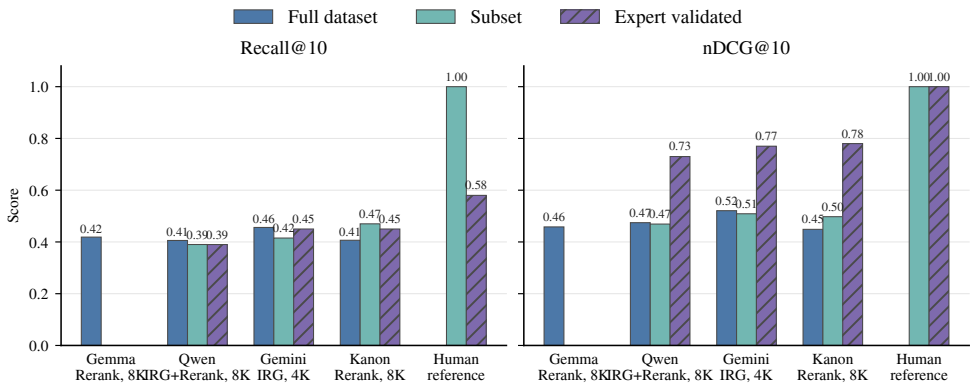

Figure 3: Retrieval metrics before and after expert evaluation of document relevance on a subset of 30 queries.

tently improves Kanon-2 across all chunk sizes and metrics – this is unsurprising given that the reranker is based on Kanon, so they are presumably designed to work well together. However, it is mostly useful for Qwen and Gemma with larger chunks – potentially because they can be noisier and reranking can help filter the noise. IRG provides the highest gains when the base model is already strong (e.g., the Gemini family). Combining IRG with reranking causes a small improvement in recall at the cost of nDCG (Fig. 2).

Expert evaluation. Fig. 3 displays the recall@10 and nDCG@10 of the best configurations before and after expert evaluation of document relevance on a subset of 30 queries. Recall marginally improved or stayed the same except for Kanon-2; Conversely, nDCG improves substantially, since high ranking documents that were considered irrelevant may now be contributing to the DCG (the numerator of nDCG). Overall, the trend suggests that a non-trivial amount of penalized system-retrieved documents were indeed relevant. In terms of comparison between models, Kanon-2 emerges as a competitive model, in contrast to the results in Table 3. However, looking at only the evaluated subset of 30 queries suggests that the automatic and expert evaluation predict a similar trend, ranking Gemini and Kanon-2 close to each other and Qwen slightly behind on both metrics. **Overall, these results suggest that automatic metrics predict a similar trend as expert evaluation, but underestimate the absolute performance of systems.**

We also re-evaluate the expert annotators from Sec. 3. Trivially, adding previously missed relevant documents leads to recall dropping significantly. This highlights the limitation of relying on experts to provide gold standard retrieval annotations, and underscores the value of iterative evaluation.

Query:
Ex won't leave my house, keeps coming back in after I tell them no. Even after police! What happens to people who keep breaking into your home like that? Is there a real penalty? How do I stop it? It's my house!

Retrieved Chunk 1 (Qwen & Kanon 2):
[omitted details...] The police, during a murder investigation in 1991, entered the accused's house (an equipment trailer) without permission. When they received no answer at the door, they entered, roused the accused, touched his leg, ordered him to get up and took him to the front of the trailer for better lighting. The police arrested him after seeing blood on his shirt. [omitted details...]

Retrieved Chunk 2 (Gemini-2):
[Omitted Details...] 3) In the stairwell of Q's apartment the accused [Omitted graphic detail...] and threatened to kill her [Omitted Details...] The offences represent a sustained pattern of physical and mental abuse of an intimate partner over a four-year period, and an effort to avoid responsibility through efforts to obstruct justice.

Table 4: Retrieved irrelevant chunks for example query.

4.5 Analysis

Case Analysis. Table 4 demonstrates a retrieval example that highlights various error cases such as retrieving documents that are topically similar to the query but differ in legal issue, party roles, and procedural posture. The query asks what happens when an ex-partner repeatedly enters your home despite being unwelcome, even after police involvement. The Qwen and Kanon 2 configurations retrieve the first chunk which discusses the police entering an accused person's home during a murder investigation, reversing the roles (police entering the home *of the accused*). The second retrieved passage concerns intimate partner violence and threats in an apartment building without discussing entrance This example motivates better retrieval methods that measure not only topical similarity, but also specific legal logic needed to answer the query.

Analysis of performance per query type. Table 5 presents the performance based on query type

Config.	Subset	Recall@10	nDCG@10	Ground.	Acc.
Gemini	Lay Person	0.461	0.505	0.747	0.419
	Legal Expert	0.451	0.537	0.784	0.492
	Ontario	0.475	0.526	0.724	0.284
	BC	0.445	0.494	0.761	0.397
	Alberta	0.428	0.519	0.755	0.380
	Other	0.467	0.538	0.822	0.475
Gemma	Lay Person	0.443	0.468	0.767	0.603
	Legal Expert	0.418	0.475	0.723	0.445
	Ontario	0.458	0.497	0.764	0.489
	BC	0.351	0.429	0.696	0.412
	Alberta	0.440	0.508	0.771	0.505
	Other	0.459	0.451	0.748	0.463
Qwen	Lay Person	0.389	0.442	0.704	0.423
	Legal Expert	0.423	0.508	0.717	0.457
	Ontario	0.438	0.514	0.743	0.475
	BC	0.387	0.463	0.701	0.426
	Alberta	0.386	0.492	0.716	0.450
	Other	0.404	0.431	0.681	0.408

Table 5: Performance per query type and jurisdiction for the best performing configuration overall for each embedding family.

(legal expert or lay person) and province. Across methods, retrieval performance is generally worse on BC and Alberta and best on Ontario, but `Gemini` has the least variation between provinces showing robustness to legal domains. Comparing the query types, the nDCG is higher for legal experts across models but recall is mixed. We compare the difference in query style performance deltas between base and enhanced configurations in Fig. 4 and Fig. 5 (Appendix D.1). In base configurations, performance is always better on queries simulating legal experts, with large gaps in nDCG@10. This illustrates how embeddings are not robust to the difference between layperson language and legal documents. Enhancements bridge the gaps in nDCG@10, likely due to IRG writing intermediate answers in legal style.

5 Generation

We present the baselines (§5.1), evaluation metrics (§5.2), and results (§5.3) for the generation task, and analyze the performance (§5.4).

5.1 Baselines and Evaluated Models

Table 14 shows the models we use for generation. We select one strong proprietary model (`Gemini Flash 2.5`) and two strong open source models (`Qwen 3.5-9B` and `Gemma 4-E4B`) for answer generation (Prompt B.7 in the appendix). We use temperature of 0.[4] Each generator is paired with retrievers based on embeddings of the same model family.

[4]We re-run some examples for `Qwen` with a temperature of 0.05 when it entered thinking loops with a temperature of 0.

5.2 Evaluation Metrics

We test the generated answers along two dimensions: (a) how similar they are to ground truth answers, and (b) how grounded they are in retrieved documents. The latter measures the inverse of hallucination rate. We use a variation of FActScore (Min et al., 2023) together with `Gemini-2.5-Pro`. We decompose the generated answer $\hat{y}_q$ into a set of atomic claims $A(\hat{y}_q) = \{a_{q,1}, \ldots, a_{q,m_q}\}$. We then look at the proportion of claims in $A(\hat{y}_q)$ that are supported by the evidence set, which is defined differently for each dimension below.

Accuracy against ground truth answers. The evidence set is $A(y_q)$, which is obtained by using the same LLM to decompose the gold standard answer y_q into atomic claims. We then compute the accuracy of claims as follows:

$$Acc(\hat{y}_q, y_q) = \frac{1}{|A(\hat{y}_q)|} \sum_{a \in A(\hat{y}_q)} \mathbb{I}\left[\max_{e \in A(y_q)} \mathrm{Supp}(a, e) = 1\right] \quad (2)$$

where $\mathrm{Supp}(a, e)$ is an entailment-based support judgment returned by the factuality evaluator (see Prompts B.8–B.11 in the appendix). A claim is counted as grounded if at least one claim in the evidence set $A(y_q)$ entails it. Following FActScore and X-FactAlign (Samir et al., 2024), this metric measures the fraction of atomic claims in the generated answer that are supported by the provided evidence, rather than independent legal correctness.

Groundedness in documents. The evidence set is E_q, under condition $c \in \{\text{ORACLE}, \text{PIPELINE}\}$. In the pipeline condition, documents are gathered from retrieval methods: $E_q = D_q^r$. In the oracle condition, we confine retrieval to only chunks from ground truth documents: $E_q^c = D_q^{GT}$.[5] The pipeline condition measures the impact that retrieval has on final answers and partially measures the ability of retrieval methods despite missing documents in the gold set. Query-level groundedness is thus similarly defined as:

$$G(q, c) = \frac{1}{|A(y_q)|} \sum_{a \in A(y_q)} \mathbb{I}\left[\max_{e \in E_q^c} \mathrm{Supp}(a, e) = 1\right] \quad (3)$$

[5]We used the Qwen Embedder 8192 chunk size with reranking configuration across oracle conditions for equal evaluation.

Cond.	LLM	n	Ground.	Acc.
Oracle	Gemini	532	0.7839	0.6693
Oracle	Gemma	532	**0.7983**	0.6351
Oracle	Qwen	526	0.7488	**0.7646**
Pipeline	Gemini	532	**0.7656**	0.3792
Pipeline	Gemma	532	0.7448	**0.4672**
Pipeline	Qwen	526	0.7103	0.4398

Table 6: Groundedness and accuracy scores across conditions. Bold numbers are best in the condition.

5.3 Results

Table 6 presents groundedness and accuracy across generations and conditions. For each model family, we used the top configuration (Table 3).

Models generate claims unsupported by documents. Across all setups, at least 20% of generated claims are unsupported by the documents.

In both conditions, `Qwen` creates the highest number of unsupported claims. This could be explained by generating longer answers averaging 14,685 characters, compared to 6057 for `Gemma` and 11,550 for `Gemini`. Qwen also produces many thinking tokens which sometimes get stuck on a wrong idea.

To investigate the nature of unsupported claims, we sampled 25 unsupported claims from the best performing model in each condition: `Gemma` in the oracle condition and `Gemini` in the pipeline condition, and categorized them in Tables 12 and 13 in Appendix, respectively. For `Gemma` oracle condition, approximately 38% of ungrounded statements are valid statements that establish the context of the query or finalize an answer, both trivially lacking from retrieved documents. Yet, 54% capture different hallucinations types such as confusing between general law vs. case specific statements, misapplication of legal tests, misattributing content between documents, and lack of sensitivity to details such as "shall not" vs. "may not". In the `Gemini-2` pipeline condition, around 60% of the claims were trivially failed, mostly from synthesizing texts and creating grounded statements too semantically far from the original texts. The other 40% of the ungrounded claims were hallucinations. Extrapolating this to results in Tab 6, estimate a lower bound of 8% hallucination rate with an upper bound of 29%.

Generated answers differ substantially from the gold standard, especially when based on different documents. The accuracy of generated answers in the oracle condition is 64-76%, demonstrating substantial deviation. We analyze unsupported claims in Appendix D.2. The accuracy is much lower in the pipeline condition, when it is based on different documents than the gold standard answer. **These findings illustrate the importance of a good retrieval component.**

5.4 Analysis

Table 5 shows that subgroup trends in generation do follow retrieval trends. Legal-expert queries have higher nDCG@10 for all three family-matched pipelines, but this does not translate uniformly into better answers. `Gemini` and `Qwen` improve on legal-expert queries for both groundedness (0.747 to 0.784, and 0.704 to 0.717) and accuracy against ground truth answers (0.419 to 0.492, and 0.423 to 0.457). `Gemma`, however, has higher groundedness and much higher accuracy for layperson queries (0.767 and 0.603) than legal-expert queries (0.723 and 0.445), despite similar nDCG@10. Thus, legal phrasing helps retrieval ranking, but improvements on generation quality remains to be seen. Our previous categorization of groundedness errors shows there can be a lot of noise in judging.

Jurisdictional results show a similar decoupling. `Gemini` is most accurate and grounded on the 'Other' subset of provinces, and performs worst on Ontario for accuracy (0.284), even with strong retrieval scores. `Gemma` is weakest on British Columbia across retrieval and generation, and strongest on Alberta for generation. `Qwen` is more stable across jurisdictions in generation. Across each configuration, there is high deviation across performance on different jurisdictions which highlights the need for more robust systems or individual systems tailored for each jurisdiction.

6 Conclusion

We introduced CANLEGALRAGBENCH, a benchmark for evaluating RAG over Canadian case law. CANLEGALRAGBENCH differs from prior benchmarks by focusing on realistic queries from users of public-facing legal assistants. Evaluation on various RAG systems reveals a high hallucinate rates, but it also highlights the limitations of automatic evaluation of relevance and accuracy. Future work should develop additional benchmarks to cover other aspects of Canadian law such as case laws in French and Quebec civil-law sources. By releasing the code and data, we aim to support transparent comparison and more reliable development of RAG systems for legal information tasks.

Limitations

Data Coverage. CANLEGALRAGBENCH only covers Canadian case law in select courts and provinces, and only focuses on English case laws. Future work can develop benchmarks that cover bilingual Canadian law, and Quebec civil-law sources, statute and regulation. In addition, due to the cost of annotations, our dataset is only meant for testing rather than training models.

Evaluation. Our findings highlighted the challenge in evaluating document relevance and answer correctness automatically. To overcome these limitations, we employed both expert evaluation and various types of analyses. However, expert annotation is costly and doesn't scale. Future work should focus on developing reliable evaluation metrics for legal RAG. Moreover, our evaluation largely ignored the temporal relevance and the precedential value of courts, which could be important to determine document relevance. We leave this challenge to future work.

Ethics Statement

Data Access. We used two sources of case laws: the CLD dataset, which is publicly available, and a private dataset provided to us by a company with permission to publish it upon acceptance. The benchmark itself combines queries generated by an LLM with answers collected from experts, and will be made publicly available upon publication. By releasing the data, prompts, model outputs, and evaluation code, we aim to support transparent comparison and more reliable development of legal RAG systems for Canadian legal information tasks.

Expert Annotation. The work was approved by our institutional review board (IRB). We hired annotators internally at our institution and through Upwork and compensated them $20 CAD per hour for students and $40 per hour for the professional paralegal. All are higher than the minimum wage. We explained to them the goal of improving automated systems for legal research and answer synthesis. To demonstrate how costly it is to collect a moderate size dataset like ours while still paying experts fairly, overall we spent around $7000 on data collection and manual evaluation. We collected consent forms from annotators where we detailed the context of our research and publication goals. We highlighted the risk of viewing sensitive content in court cases which they agreed to.

Implications. Performance on CANLEGALRAGBENCH is not directly indicative of real-world performance, and we do not advocate for replacing legal expertise with AI. People should still consult lawyers when they face legal issues, and lawyers are still responsible for the quality and accuracy of their legal research, whether or not it was supported by AI tools.

Privacy Case text contains names and other personal information however we do not make public any private data or information that was not previously public.

License

We will make available our annotations and code under an open MIT license. Data from courts are subject to their own licenses which have limitations for commercial use.

References

David Beauchemin, Richard Khoury, and Zachary Gagnon. 2024. Quebec automobile insurance question-answering with retrieval-augmented generation. In *Proceedings of the Natural Legal Language Processing Workshop 2024*, pages 48–60, Miami, FL, USA. Association for Computational Linguistics.

Umar Butler, Abdur-Rahman Butler, and Adrian Lucas Malec. 2025. The massive legal embedding benchmark (mleb). *Preprint*, arXiv:2510.19365.

Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael Bommarito, Ion Androutsopoulos, Daniel Katz, and Nikolaos Aletras. 2022. LexGLUE: A benchmark dataset for legal language understanding in English. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4310–4330, Dublin, Ireland. Association for Computational Linguistics.

Andong Chen, Feng Yao, Xinyan Zhao, Yating Zhang, Changlong Sun, Yun Liu, and Weixing Shen. 2023. Equals: A real-world dataset for legal question answering via reading chinese laws. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Law*, ICAIL '23, page 71–80, New York, NY, USA. Association for Computing Machinery.

Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, page 758–759, New York, NY, USA. Association for Computing Machinery.

Matthew Dahl, Varun Magesh, Mirac Suzgun, and Daniel E Ho. 2024. Large legal fictions: Profiling legal hallucinations in large language models. *Journal of Legal Analysis*, 16(1):64–93.

Zhiwei Fei, Xiaoyu Shen, Dawei Zhu, Fengzhe Zhou, Zhuo Han, Alan Huang, Songyang Zhang, Kai Chen, Zhixin Yin, Zongwen Shen, Jidong Ge, and Vincent Ng. 2024. LawBench: Benchmarking legal knowledge of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7933–7962, Miami, Florida, USA. Association for Computational Linguistics.

Neel Guha, Julian Nyarko, Daniel Ho, Christopher Ré, Adam Chilton, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel Rockmore, Diego Zambrano, and 1 others. 2023. Legalbench: A collaboratively built benchmark for measuring legal reasoning in large language models. *Advances in neural information processing systems*, 36:44123–44279.

Ben Hagag, Gil Gil Semo, Dor Bernsohn, Liav Harpaz, Pashootan Vaezipoor, Rohit Saha, Kyryl Truskovskyi, and Gerasimos Spanakis. 2024. LegalLens shared task 2024: Legal violation identification in unstructured text. In *Proceedings of the Natural Legal Language Processing Workshop 2024*, pages 361–370, Miami, FL, USA. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. 2021. CUAD: An expert-annotated nlp dataset for legal contract review. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Abe Bohan Hou, Orion Weller, Guanghui Qin, Eugene Yang, Dawn Lawrie, Nils Holzenberger, Andrew Blair-Stanek, and Benjamin Van Durme. 2025. CLERC: A dataset for U. S. legal case retrieval and retrieval-augmented analysis generation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 7913–7928, Albuquerque, New Mexico. Association for Computational Linguistics.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

Prathamesh Kalamkar, Astha Agarwal, Aman Tiwari, Smita Gupta, Saurabh Karn, and Vivek Raghavan. 2022. Named entity recognition in Indian court judgments. In *Proceedings of the Natural Legal Language Processing Workshop 2022*, pages 184–193, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2023. Dspy: Compiling declarative language model calls into self-improving pipelines. *Preprint*, arXiv:2310.03714.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Jonathan Li, Rohan Bhambhoria, Samuel Dahan, and Xiaodan Zhu. 2024. Experimenting with legal ai solutions: The case of question-answering for access to justice. *arXiv preprint arXiv:2409.07713*.

Antoine Louis and Gerasimos Spanakis. 2022. A statutory article retrieval dataset in French. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6789–6803, Dublin, Ireland. Association for Computational Linguistics.

Yuanhua Lv and ChengXiang Zhai. 2011. Lower-bounding term frequency normalization. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, page 7–16, New York, NY, USA. Association for Computing Machinery.

Robert Mahari, Dominik Stammbach, Elliott Ash, and Alex Pentland. 2024. LePaRD: A large-scale dataset of judicial citations to precedent. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9863–9877, Bangkok, Thailand. Association for Computational Linguistics.

Mihai Masala, Radu Cristian Alexandru Iacob, Ana Sabina Uban, Marina Cidota, Horia Velicu, Traian Rebedea, and Marius Popescu. 2021. jurBERT: A Romanian BERT model for legal judgement prediction. In *Proceedings of the Natural Legal Language Processing Workshop 2021*, pages 86–94, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Robert R. McCrae and Oliver P. John. 1992. An introduction to the five-factor model and its applications. *Journal of Personality*, 60(2):175–215.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

Joel Niklaus, Ilias Chalkidis, and Matthias Stürmer. 2021. Swiss-judgment-prediction: A multilingual legal judgment prediction benchmark. In *Proceedings of the Natural Legal Language Processing Workshop 2021*, pages 19–35, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Vasile Pais, Maria Mitrofan, Carol Luca Gasan, Vlad Coneschi, and Alexandru Ianov. 2021. Named entity recognition in the Romanian legal domain. In *Proceedings of the Natural Legal Language Processing Workshop 2021*, pages 9–18, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Nhi Pham, Lachlan Pham, and Adam L. Meyers. 2021. Legal terminology extraction with the termolator. In *Proceedings of the Natural Legal Language Processing Workshop 2021*, pages 155–162, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Nicholas Pipitone and Ghita Houir Alami. 2024. Legalbench-rag: A benchmark for retrieval-augmented generation in the legal domain. *arXiv preprint arXiv:2408.10343*.

Juliano Rabelo, Randy Goebel, Mi-Young Kim, Yoshinobu Kano, Masaharu Yoshioka, and Ken Satoh. 2022. Overview and discussion of the competition on legal information extraction/entailment (coliee) 2021. *The Review of Socionetwork Strategies*, 16(1):111–133.

Dragomir R. Radev, Hong Qi, Harris Wu, and Weiguo Fan. 2002. Evaluating web-based question answering systems. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, Las Palmas, Canary Islands - Spain. European Language Resources Association (ELRA).

Abhilasha Ravichander, Alan W Black, Shomir Wilson, Thomas Norton, and Norman Sadeh. 2019. Question answering for privacy policies: Combining computational and legal perspectives. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4947–4958, Hong Kong, China. Association for Computational Linguistics.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at trec-3. In *TREC*, volume 500-225 of *NIST Special Publication*, pages 109–126. National Institute of Standards and Technology (NIST).

Farhan Samir, Chan Young Park, Anjalie Field, Vered Shwartz, and Yulia Tsvetkov. 2024. Locating information gaps and narrative inconsistencies across languages: A case study of LGBT people portrayals on Wikipedia. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6747–6762, Miami, Florida, USA. Association for Computational Linguistics.

Gil Semo, Dor Bernsohn, Ben Hagag, Gila Hayat, and Joel Niklaus. 2022. ClassActionPrediction: A challenging benchmark for legal judgment prediction of class action cases in the US. In *Proceedings of the Natural Legal Language Processing Workshop 2022*, pages 31–46, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274, Singapore. Association for Computational Linguistics.

Dan Simonson. 2021. Supervised identification of participant slots in contracts. In *Proceedings of the Natural Legal Language Processing Workshop 2021*, pages 163–171, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Santosh T.y.s.s., Rashid Haddad, and Matthias Grabmair. 2024. ECtHR-PCR: A dataset for precedent understanding and prior case retrieval in the European court of human rights. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 5473–5483, Torino, Italia. ELRA and ICCL.

Henrique Schechter Vera, Sahil Dua, Biao Zhang, Daniel Salz, Ryan Mullins, Sindhu Raghuram Panyam, Sara Smoot, Iftekhar Naim, Joe Zou, Feiyang Chen, Daniel Cer, Alice Lisak, Min Choi, Lucas Gonzalez, Omar Sanseviero, Glenn Cameron, Ian Ballantyne, Kat Black, Kaifeng Chen, and 70 others. 2025. Embeddinggemma: Powerful and lightweight text representations. *Preprint*, arXiv:2509.20354.

Simon Wallace and Sean Rehaag. 2025. Introducing the a2aj's canadian legal data: An open-source alternative to canlii for the era of computational law. *Preprint*, arXiv:2509.13032.

Steven Wang, Antoine Scardigli, Leonard Tang, Wei Chen, Dmitry Levkin, Anya Chen, Spencer Ball, Thomas Woodside, Oliver Zhang, and Dan Hendrycks. 2023. MAUD: An expert-annotated legal NLP dataset for merger agreement understanding. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16369–16382, Singapore. Association for Computational Linguistics.

Mohr Wenger, Tom Kalir, Noga Berger, Carmit Klar Chalamish, Renana Keydar, and Gabriel Stanovsky. 2021. Automated extraction of sentencing decisions from court cases in the Hebrew language. In *Proceedings of the Natural Legal Language Processing Workshop 2021*, pages 36–45, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *Preprint*, arXiv:2506.05176.

Lucia Zheng, Neel Guha, Javokhir Arifov, Sarah Zhang, Michal Skreta, Christopher D. Manning, Peter Henderson, and Daniel E. Ho. 2025. A reasoning-focused

legal retrieval benchmark. In *CSLAW '25 (forthcoming)*.

A Annotation Details

We accepted students who had above-average grades and/or prior work experience under a lawyer. Annotators were provided a UI that presented all the queries for a given seed query. They were instructed to first flag and skip nonsensical, ill-formed or very complex queries. Then, they were tasked with searching for relevant case laws. We allowed annotators to use common search tools that they would normally use in legal research, such as the Canadian Legal Information Institute (CanLii) and Thomson Reuters' Westlaw.[6] For each case law they retrieved, they were asked to select relevant text of each document. After completing the retrieval stage, they moved on to the answer phase. To speed up the annotation process, instead of asking annotators to phrase an answer from scratch, we provided the query and the selected relevant texts to `Gemini 2.5 Pro` to generate an answer, and instructed the annotators to verify or edit to a final answer. This design was informed by authors who are legal experts, who indicated that searching for relevant case laws is the most challenging step of legal research. To prevent AI influence in the retrieval stage, once the annotators moved on to the answer generation stage, the retrieval stage was locked.

The following are the original instructions given to instructors. We received feedback from the annotators and emailed clarifications and updates throughout. For example, part way through, we lifted the max document limit while stating they should still aim to include less than 10 documents per query on average. We also allowed them to use statutes part way through to improve the speed, though we still required at least 1 case law per query.

[6] While the search in these tools may involve AI, annotators were still responsible for constructing the queries and manually selecting among the retrieved cases. To improve the completeness of the final answers, we also allowed them to also cite laws, but we required that each query have at least one case law.

Original Annotation Guidelines

```
Load a Batch: Enter a batch ID below. Each batch contains 8 unique legal queries. There are 2 fact
variations per batch. Each has 4 situation variations which aim to keep the base facts the same.
Track Your Time: Before starting each session, make sure to start a timer. We use this to track how
long it takes to complete each batch so we can pay you appropriately!
Evaluate the Query: Read the queries and evaluate them using the provided form. If a query is too
vague or too complex, you can flag it and skip it. If it is obviously too complex to be completed
within a reasonable time, you can skip it. If more than 2 queries are flagged, just skip the batch.
After doing a few, you can get a sense of how long each one should take and adjust your approach
accordingly.
Gather Evidence: Read each query and use external resources (WestLaw, CanLII, etc.) to find relevant
cases for the given jurisdiction. Only use court cases or regulations. Add your source documents to
the query, including the case title, citation, and URL.

Target Scope:
Every 4 queries should take around 40-80 minutes (less is great). If it consistently takes longer, use
 fewer sources. Do not use more than 10 sources (pick the best ones). Do not include excessively long
documents that take a long time to read if possible (use your judgement). We are targeting to have
each student complete around 4-8 batches per week.
Extract & Paste: Read your sources and copy the highly relevant quotes into the "Snippets" section (
one point per snippet). Copy the whole contents of the original source and paste it into the "Whole
Document Text" field. If there isn't a good way to get the text, just do `Ctrl+A` and adjust so that
the main text is selected. Do not spend much time on copying or formatting the sources!
Reuse Work: Use the Copy to... menu on a document to instantly duplicate it across multiple queries.
AI Synthesis & Final Answer Standards: Use the Custom Instructions box to guide the AI's focus. Click
either "Generate (Snippets Only)" or "Generate (Include Full Text)". If the full text is too long, the
 tool will warn you to use Snippets Only.

Important: Document Locking
Generating an AI answer will permanently lock your documents for that query to prevent biasing your
research based on the AI's output. You can still read the documents, but you cannot edit, add, or
delete them after generation. Make sure your research is complete and you are ready before generating!

What makes a good Final Answer?
Required structure - three sections:
Opening Statements: Introduce the topic/area of law, paraphrase the question to make the legal issue
clear, give a short hedge of the conclusion.
Supporting Arguments: Arguments and evidence drawn from the gathered documents, with discussion of how
 they support or qualify the answer.
Final Conclusion: A clear concluding statement synthesising the above.
Synthesize, don't just answer: Focus on what *previously happened* in court, rather than giving a
definitive "yes" or "no".
Acknowledge Missing Facts: If important query details are NOT covered by your evidence, explicitly
state this and explain why those missing facts might matter. For example:
"Based on the provided evidence, it appears that courts determine that factor X was the most important
 in reaching the outcome (2022 ONCA 45). However, there is no evidence about factor Y, which could
also be important because..."
"The most relevant pieces of evidence are (2022 ONCA 45) which says X and (2019 BCCA 12) which says Y.
 Based on your case, it's not clear which way it could go     some things you should keep in mind are
..."
Exact Citations: Citations must appear exactly as entered in the Citation field. If a citation cannot
be verified quickly, use neutral citation format (e.g. Case Name, 2022 ONCA 45) or remove it.
Stay Grounded: Rely ONLY on the provided evidence and very general legal knowledge. Omit introductory
filler.
Crucial Step: The AI is just a drafter. You must read, edit, and manually verify the generated answer
to ensure it meets these standards and accurately satisfies the query.
```

B Prompt Templates

Prompt B.1: Query Generation Example

```
Your input fields are:
1. 'court_decision_text' (str): A full Canadian legal court decision.
2. 'user_persona' (str): The type of user using a Legal AI tool and their intent (e.g., 'Layperson looking
for the outcome', 'Lawyer looking for citations').
3. 'user_traits' (str): A description of the user's personality.
4. 'target_section' (str): The section of the document to focus on: Overview, Reasoning, Decision.

Your output fields are:
1. 'generated_query' (str): A realistic search query that the user might type into a Legal AI tool,
based on their persona and the target section.

All interactions will be structured in the following way, with the appropriate values filled in.

[[ ## court_decision_text ## ]]
{court_decision_text}

[[ ## user_persona ## ]]
{user_persona}

[[ ## user_traits ## ]]
{user_traits}

[[ ## target_section ## ]]
{target_section}

[[ ## generated_query ## ]]
{generated_query}

[[ ## completed ## ]]

In adhering to this structure, your objective is:
Given a court decision and a specific user persona/intent and target section of the document (Overview,
Reasoning, Decision), generate a realistic search query that a user might type, and the corresponding
factual answer found strictly within the document around the target section.

The generated query must be a realistic search query or question relevant to the text but assuming no
knowledge of the document. The query should be asked from the point of view of a user doing exploratory
research on their issue. Do NOT reference specific names, dates, entities, or other specific case details.
Create queries that a user might ask *without* knowing which case document contains the answer. The
queries should reflect the legal issues, principles, fact patterns, and situations discussed in the
provided court decision, but MUST NOT reveal or directly quote the document.

The Query must have at least 2-3 unique facts or distinguishing details that are relevant to the document
but do not directly reveal the document. For example, if the document is about a car accident case,
the query could be "What happens if on a snowy day where the conditions were poor, I'm involved in a
rear-end collision and the other driver was texting?"

Introduce variety in the types of queries generated, including but not limited to:
- Hypothetical situations ("If I ...", "What happens when ...?")
- Conceptual legal query (How does the court determine...?")
- Outcome-oriented query ("Would this count as...?")
- Multi-step or scenario-based query requiring reasoning.
- The query should be moderately complex, not trivial.

[[ ## court_decision_text ## ]]
<FULL COURT DECISION TEXT GOES HERE>

[[ ## user_persona ## ]]
Layperson: A non-legal expert asking a general query about a realistic legal scenario they may face in the
real world. They do not use legal jargon.

[[ ## user_traits ## ]]
User traits: openness: low, conscientiousness: medium, extraversion: high, agreeableness: high,
neuroticism: high.

Query Style Instructions: Do NOT cite specific case names, statute numbers, or legal tests. Use perfect
grammar, precise legal terminology, and complex sentence structures. Use standard, functional grammar.
Use loose grammar, potential typos, simple keywords, or fragment sentences. Frame the query as a 'rambling'
or 'venting' personal story. Include specific details of what went wrong to give context, rather than
just asking the legal question directly. Be terse, brief, and to the point. Tone should be urgent,
worried, seeking reassurance, or focusing on risks/penalties. Tone should be calm, detached, and
objective. Focus on abstract concepts, reasoning, implications, or 'why' questions. Focus strictly on the
tangible 'Adverse Event' or 'Injury' described in the text (e.g., 'forced to retake class', 'denied
refund', 'basement flooded'). Do not use abstract summaries like 'school problem' or 'career help'.
Use polite, soft openers and tone (e.g., 'I'm hoping you can help...'), but ensure the core complaint
describes the specific unfair event mentioned in the text. Phrasing should be demanding, skeptical, or
aggressive (e.g., 'Prove that...').

[[ ## target_section ## ]]
Reasoning

Respond with the corresponding output fields, starting with the field  [[ ## generated_query ## ]], and
then ending with the marker for  [[ ## completed ## ]].
```

Prompt B.2: Query Variations - Generate Alternative Facts

```
=== Message 0 | Role: system ===
Your input fields are:
1. `original_query` (str): The original query for which to generate a fact
variation.
Your output fields are:
1. `reasoning` (str):
2. `query_variation` (str): The generated query variation based on the input
original query with altered facts but the same scenario type and tone/style.
All interactions will be structured in the following way, with the appropriate
values filled in.

[[ ## original_query ## ]]
{original_query}

[[ ## reasoning ## ]]
{reasoning}

[[ ## query_variation ## ]]
{query_variation}

[[ ## completed ## ]]
In adhering to this structure, your objective is:
Creating queries to evaluate a RAG tool that helps everyday people navigate legal
issues.
Receives a query and generates a variation on the facts of the query while keeping
 the same scenario type and tone/style. The fact variation should be a realistic
alteration of the original facts that still fits within the same general scenario
type.
The query variation should change what court cases would be relevant to the query,
 but should not be so outlandish as to change the scenario type. The variations
should also not change the tone and style of the query.
The query variation should not increase the complexity of the query or make it
more difficult to understand. The query variation should be something that a
person might realistically ask about in relation to the same type of legal issue.

Examples:
1. Original query: "My kid, he's 8, got into a fight and punched someone just one
time. But the person got seriously hurt. Now we're dealing with court. Will he go
to jail? Like, what kind of punishment could he get for causing bad injuries from
one punch? I'm so worried.""
Fact variation example: ""My kid, he's 17, got into a fight and punched someone
just one time. But the person got seriously hurt. Now we're dealing with court.
Will he go to jail? Like, what kind of punishment could he get for causing bad
injuries from one punch? I'm so worried.""

2. Original query: "My ex keeps making up stories that I hurt our child, even when
 all the investigations show nothing happened. It's so frustrating because they
keep involving child services and now everyone thinks *her* constant talk about
abuse is actually making our child suffer emotionally. What happens if one parent
won't stop pushing false allegations and it causes harm to the child, and what can
 be done to protect the child from their other parent's actions like that?"
Fact variation example: "My ex keeps making up stories that I hurt our child. It's
 so frustrating because they keep involving child services and now everyone thinks
 *my* constant talk about abuse is actually making our child suffer emotionally.
What happens if one parent won't stop pushing false allegations and it causes harm
 to the child, and what can be done to protect the child from their other parent's
 actions like that?"

=== Message 1 | Role: user ===
[[ ## original_query ## ]]
{query}Respond with the corresponding output fields, starting with the field `[[
## reasoning ## ]]`, then `[[ ## query_variation ## ]]`, and then ending with the
marker for `[[ ## completed ## ]]`.
```

Prompt B.3: Query Variations - Generate Alternative Situation

```
=== Message 0 | Role: system ===
Your input fields are:
1. `query` (str): The original query for which to generate variations.
2. `query_variation_type` (str): The type of variation to create
3. `variation_description` (str): A description of the type of variation to create
4. `query_variation_example` (str): An example of the type of variation to create
5. `original_query_example` (str): An example of an original query that the variation example is based on
6. `query_variation_tone_style` (str): The tone and style to use when creating the query variation.
For example, 'layperson' or 'legal expert'.
Your output fields are:
1. `reasoning` (str):
2. `query_variation` (str): The generated query variation based on the input query, variation type,
and tone/style.
All interactions will be structured in the following way, with the appropriate values filled in.

[[ ## query ## ]]
{query}

[[ ## query_variation_type ## ]]
{query_variation_type}

[[ ## variation_description ## ]]
{variation_description}

[[ ## query_variation_example ## ]]
{query_variation_example}

[[ ## original_query_example ## ]]
{original_query_example}

[[ ## query_variation_tone_style ## ]]
{query_variation_tone_style}

[[ ## reasoning ## ]]
{reasoning}

[[ ## query_variation ## ]]
{query_variation}

[[ ## completed ## ]]
In adhering to this structure, your objective is:
Creating queries to evaluate a RAG tool that helps everyday people navigate legal issues.
The queries will be annotated with legal experts and are thus costly to annotate. To reduce cost,
the variations should require minimal changes to the court case documents that are required to answer the
query. The variations should not be so outlandish as to create unrealistic queries that people would never
actually ask about legal issues. The variations should also not change the tone and style of the query
unless specified to be in the tone of a legal expert.

Receives a query and variation type and tone/style. Then creates a variation of the query.

=== Message 1 | Role: user ===
[[ ## query ## ]]
{query}

[[ ## query_variation_type ## ]]
{variation_type}

[[ ## variation_description ## ]]
{variation_description}

[[ ## query_variation_example ## ]]
{variation_example}

[[ ## original_query_example ## ]]
What are my rights if my landlord enters my apartment without notice?

[[ ## query_variation_tone_style ## ]]
{layperson/legal expert}

Respond with the corresponding output fields, starting with the field `[[ ## reasoning ## ]]`,
then `[[ ## query_variation ## ]]`, and then ending with the marker for `[[ ## completed ## ]]`.
```

Prompt B.4: LLM Judge for Query Creation - System Prompt

```
Your input fields are:
1. 'document_snippet' (str): The legal text serving as the ground truth answer.
2. 'query' (str): The generated user question to be evaluated.
3. 'criteria_guidelines' (str): The specific rules and examples this pair must
satisfy.
Your output fields are:
1. 'reasoning' (str):
2. 'analysis' (str): A step-by-step analysis of how the pair meets or fails EACH
criterion.
3. 'final_verdict' (str): Must be exactly 'Keep' or 'Reject'.
All interactions will be structured in the following way, with the appropriate
values filled in.

[[ ## document_snippet ## ]]
{document_snippet}

[[ ## query ## ]]
{query}

[[ ## criteria_guidelines ## ]]
{criteria_guidelines}

[[ ## reasoning ## ]]
{reasoning}

[[ ## analysis ## ]]
{analysis}

[[ ## final_verdict ## ]]
{final_verdict}

[[ ## completed ## ]]
In adhering to this structure, your objective is:
You are an expert Legal Retrieval Judge.
You will audit a (Query, Document) pair to decide if it belongs in a Gold Standard
Test Set for a realistic Legal Retrieval system that retrieves documents based on
questions people have about the law or legal precedents.
The QUERY is a user query that should be answerable by retrieving the DOCUMENT.
The DOCUMENT is a legal court decision that serves as the ground truth document to
retrieve in a Legal Retrieval system.

You must evaluate the pair against the provided CRITERIA GUIDELINES.
If the pair fails ANY of the criteria, the final verdict must be 'Reject'.
```

Prompt B.5: LLM Judge for Query Creation - User Prompt

```
[[ ## document_snippet ## ]]
<the actual document text>

[[ ## query ## ]]
<the actual user query>

[[ ## criteria_guidelines ## ]]
### Structural Integrity (No AI Artifacts)
**Rule**: The query must not contain 'AI-isms', references to 'the provided text',
 or explicit request for summaries. It should sound like a human asking a question
, not a prompt engineer.
**Examples**:
- [FAIL]: Summarize the provided legal document regarding the plaintiff.
- [FAIL]: Based on the text below, what was the ruling?
- [FAIL]: Extract the key dates from this case file.
- [PASS]: What is the precedent for constructive dismissal in Ontario?
- [PASS]: Can a landlord evict a tenant for personal use if they own another
property?

### Information Gap (The 'Google Test')
**Rule**: The query must be specific enough to require a legal retrieval tool. If
the
query is trivial generic knowledge (answerable by a quick Google search without
this
doc), it fails.
**Examples**:
- [FAIL]: What does the Supreme Court do?
- [FAIL]: Define 'contract'.
- [FAIL]: Who is the current Prime Minister?
- [PASS]: Does the Oakes test apply to administrative tribunals?
- [PASS]: limitations period for medical malpractice in British Columbia 2023

### Relevance & Grounding
**Rule**: The document must actually contain the answer or be highly relevant
precedent
for the query. The query must not hallucinate facts not present in the document.
**Examples**:
- [FAIL]: Query asks about a 'drunk truck driver' when the document is about a '
slip and fall' in a grocery store.
- [FAIL]: Query asks for the 2024 statute amendment, but the document is a 1990
case decision.
- [PASS]: Query asks about 'slip and fall liability' and the document is a ruling
on 'Doe v. Walmart' regarding a wet floor.
- [PASS]: Query asks for 'grounds for patent invalidation' and the document
outlines a decision on 'obviousness' in a patent suit.

### Tax court specific test for relevant cases
**Rule**: The document must be about self represented litigants, informal
procedures, tax dispute resolution mechanisms or other disputes that are relevant
to everyday
people. This is specifically about the type of query and not about matching the
query to the document. If the document is about a tax court case that is very
technical and not relevant to everyday people, it fails.
**Examples**:
- [FAIL]: The document is about a complex corporate tax dispute between two
multinational corporations that has no relevance to everyday people.
- [FAIL]: The document is about a tax court case that is very technical and not
relevant to everyday people.
- [PASS]: The document is about a tax court case involving a self represented
litigant disputing a CRA assessment for a small business.
- [PASS]: The document is about a tax court case involving an individual disputing
 the disallowance of a home office deduction.

Respond with the corresponding output fields, starting with the field [[ ##
reasoning ## ]], then [[ ## analysis ## ]], then [[ ## final_verdict ## ]], and
then ending with the marker for [[ ## completed ## ]].
```

Prompt B.6: Iter-RetGen

```
System:
You are a legal research assistant drafting an intermediate answer that will be
used to refine the next retrieval round.
Answer the question accurately and concisely using ONLY the provided context
passages. Focus on what courts have decided previously rather than issuing a
definitive yes / no. If the passages do not contain enough information for some
aspect of the question, say so clearly and name the missing aspect so the next
search can target it. Cite the relevant passage(s) when possible.

User:
Context:
[1] ({citation})
{chunk_text}

[2] ({citation})
{chunk_text}

...

Question: {query_text}
```

Prompt B.7: Answer Generation

```
You are a legal research assistant answering questions about Canadian law
using ONLY the provided context passages and very general legal knowledge.
Do not focus on giving a definitive 'yes' or 'no' answer. Synthesise the
evidence into a clear, concise response that describes what courts have
previously decided on similar issues.
If important details in the question are NOT covered by the passages, state
explicitly that the evidence is insufficient for those aspects and explain
why those missing facts could matter. Do not invent information to fill gaps.
Structure your answer in exactly three sections using these plain-text headings
(no markdown):
1. Opening Statements
- Introduce the topic and general area of law.
- Paraphrase the question to make the legal issue clear.
- Give a short hedge of the conclusion.
2. Supporting Arguments
- Arguments and evidence drawn from the provided passages.
- Discussion of how the evidence supports or qualifies the answer.
3. Final Conclusion
- A clear concluding statement synthesising the above.
CITATION FORMAT: Cite sources using the exact citation string shown in each
passage header (e.g. '2022 ONCA 45'). Do not include the case name, paragraph,
section, or page references. Do not paraphrase or invent citations. If no
citation is available, omit the reference.
Omit introductory filler.

Context:
{context}

Question: {query}
```

Prompt B.8: Response Groundedness

```
You are a world class expert designed to evaluate the groundedness of an assertion
.
You will be provided with an assertion and a context.
Your task is to determine if the assertion is supported by the context.
Follow the instructions below:
A. If there is no context or no assertion or context is empty or assertion is
empty, say 0.
B. If the assertion is not supported by the context, say 0.
C. If the assertion is partially supported by the context, say 1.
D. If the assertion is fully supported by the context, say 2.
You must provide a rating of 0, 1, or 2, nothing else.
Return your response as JSON in this format: {"rating": X} where X is 0, 1, or 2.
Please return the output in a JSON format that complies with the following schema
as specified in JSON Schema:
{"description": "Structured output for response groundedness evaluation.", "
properties": {"rating": {"description": "Groundedness rating (0, 1, or 2)", "title
": "Rating", "type": "integer"}}, "required": ["rating"], "title": "
ResponseGroundednessOutput", "type": "object"}Do not use single quotes in your
response but double quotes,properly escaped with a backslash.

--------EXAMPLES-----------
Example 1
Input: {
    "response": "Albert Einstein was born in Germany.",
    "context": "Albert Einstein was born March 14, 1879 at Ulm, in Wurttemberg,
    Germany."
}
Output: {
    "rating": 2
}

Example 2
Input: {
    "response": "Einstein was a chemist who invented gunpowder.",
    "context": "Albert Einstein was a theoretical physicist known for his theory
    of relativity."
}
Output: {
    "rating": 0
}

Example 3
Input: {
    "response": "Einstein received the Nobel Prize.",
    "context": "Albert Einstein received the 1921 Nobel Prize in Physics for his
    services to theoretical physics."
}
Output: {
    "rating": 2
}
-----------------------------

Now perform the same with the following input
input: {
    "response": "<GENERATED_ANSWER>",
    "context": "<GOLD_ANSWER>"
}
Output:
```

Prompt B.9: Response Groundedness Judge

```
As a specialist in assessing the strength of connections between statements and
their given contexts, I will evaluate the level of support an assertion receives
from the provided context. Follow these guidelines:

* If the assertion is not supported or context is empty or assertion is empty,
assign a score of 0.
* If the assertion is partially supported, assign a score of 1.
* If the assertion is fully supported, assign a score of 2.

I will provide a rating of 0, 1, or 2, without any additional information.
Return your response as JSON in this format: {"rating": X} where X is 0, 1, or 2.
Please return the output in a JSON format that complies with the following schema
as specified in JSON Schema:
{"description": "Structured output for response groundedness evaluation.", "
properties": {"rating": {"description": "Groundedness rating (0, 1, or 2)", "title
": "Rating", "type": "integer"}}, "required": ["rating"], "title": "
ResponseGroundednessOutput", "type": "object"}Do not use single quotes in your
response but double quotes,properly escaped with a backslash.

--------EXAMPLES-----------
Example 1
Input: {
    "response": "Albert Einstein was a scientist.",
    "context": "Albert Einstein was a German-born theoretical physicist widely
    held to be one of the greatest and most influential scientists of all time."
}
Output: {
    "rating": 2
}

Example 2
Input: {
    "response": "Einstein invented television.",
    "context": "Albert Einstein developed the theory of relativity."
}
Output: {
    "rating": 0
}

Example 3
Input: {
    "response": "Einstein won a Nobel Prize.",
    "context": "Albert Einstein received the 1921 Nobel Prize in Physics."
}
Output: {
    "rating": 2
}
-----------------------------

Now perform the same with the following input
input: {
    "response": "<GENERATED_ANSWER>",
    "context": "<GOLD_ANSWER>"
}
Output:
```

Prompt B.10: Claim Decomposition

```
Decompose and break down each of the input sentences into one or more standalone
statements. Each statement should be a standalone claim that can be independently
verified.
Follow the level of atomicity and coverage as shown in the examples.
Please return the output in a JSON format that complies with the following schema
as specified in JSON Schema:
{"description": "Output from claim decomposition.", "properties": {"claims": {"
description": "Decomposed claims", "items": {"type": "string"}, "title": "Claims",
 "type": "array"}}, "required": ["claims"], "title": "ClaimDecompositionOutput", "
type": "object"}Do not use single quotes in your response but double quotes,
properly escaped with a backslash.

--------EXAMPLES-----------
Example 1
Input: {
    "response": "Charles Babbage was a French mathematician, philosopher, and food
     critic.",
    "atomicity": "low",
    "coverage": "low"
}
Output: {
    "claims": [
        "Charles Babbage was a mathematician and philosopher."
    ]
}

Example 2
Input: {
    "response": "Albert Einstein was a German theoretical physicist. He developed
    the theory of relativity and also contributed to the development of quantum
    mechanics.",
    "atomicity": "low",
    "coverage": "low"
}
Output: {
    "claims": [
        "Albert Einstein was a German physicist.",
        "Albert Einstein developed relativity and contributed to quantum mechanics
        ."
    ]
}
-----------------------------

Now perform the same with the following input
input: {
    "response": "<GENERATED_ANSWER>",
    "atomicity": "low",
    "coverage": "low"
}
Output:
```

Prompt B.11: Entailment-based Support Judgement

```
Your task is to judge the faithfulness of a series of statements based on a given context. For each
statement you must return verdict as 1 if the statement can be directly inferred based on the context or 0
if the statement can not be directly inferred based on the context.
Please return the output in a JSON format that complies with the following schema as specified in JSON
Schema:
{"\$defs": {"StatementFaithfulnessAnswer": {"description": "Individual statement with reason and verdict
for NLI evaluation.", "properties": {"statement": {"description": "the original statement, word-by-word",
"title": "Statement", "type": "string"}, "reason": {"description": "the reason of the verdict", "title":
"Reason", "type": "string"}, "verdict": {"description": "the verdict(0/1) of the faithfulness.", "title":
"Verdict", "type": "integer"}}, "required": ["statement", "reason", "verdict"], "title":
"StatementFaithfulnessAnswer", "type": "object"}}, "description": "Structured output for NLI statement
evaluation.", "properties": {"statements": {"items": {"\$ref": "#/\$defs/StatementFaithfulnessAnswer"},
"title": "Statements", "type": "array"}}, "required": ["statements"], "title": "NLIStatementOutput",
"type": "object"}

Do not use single quotes in your response but double quotes, properly escaped with a backslash.

--------EXAMPLES-----------
Example 1
Input: {
    "context": "John is a student at XYZ University. He is pursuing a degree in Computer Science. He is
    enrolled in several courses this semester, including Data Structures, Algorithms, and Database
    Management. John is a diligent student and spends a significant amount of time studying and completing
    assignments. He often stays late in the library to work on his projects.",
    "statements": [
        "John is majoring in Biology.",
        "John is taking a course on Artificial Intelligence.",
        "John is a dedicated student.",
        "John has a part-time job."
    ]
}
Output: {
    "statements": [
        {
            "statement": "John is majoring in Biology.",
            "reason": "John's major is explicitly stated as Computer Science, not Biology.",
            "verdict": 0
        },
        {
            "statement": "John is taking a course on Artificial Intelligence.",
            "reason": "The context mentions courses in Data Structures, Algorithms, and Database
            Management, but does not mention Artificial Intelligence.",
            "verdict": 0
        },
        {
            "statement": "John is a dedicated student.",
            "reason": "The context states that John is a diligent student who spends a significant amount
            of time studying and completing assignments.",
            "verdict": 1
        },
        {
            "statement": "John has a part-time job.",
            "reason": "There is no information in the context about John having a part-time job.",
            "verdict": 0
        }
    ]
}
----------------------------

Now perform the same with the following input
input: {
    "context": "<CONTEXT>",
    "statements": [
        "<CLAIM_1>",
        "<CLAIM_2>",
        "...",
        "<CLAIM_N>"
    ]
}
Output:
```

C Additional Information

C.1 Court Statistics

Court	Proportion	Court	Proportion	Court	Proportion
BCSC	11.55%	ONSC	10.12%	SCC	9.49%
UNKNOWN	9.31%	TCC	6.80%	STATUTE	4.66%
CHRT	4.66%	SST	4.48%	ABQB	3.40%
FCA	3.31%	BCCA	3.04%	FCT	2.33%
ONCJ	2.24%	ABPC	1.88%	ABCA	1.79%
HRTO	1.61%	FC	1.52%	BCPC	1.52%
SKCA	1.52%	ONCA	1.25%	NSSC	1.25%
SKQB	1.07%	AHRC	0.90%	NWTSC	0.81%
NBQB	0.72%	NLSC	0.63%	NLTD	0.63%
ABKB	0.54%	QCCQ	0.54%	NLCA	0.45%
NLSCTD	0.45%	NBKB	0.45%	QCCS	0.45%
ONSCDC	0.45%	QCCA	0.36%	NBCA	0.36%
NSCA	0.36%	PESCTD	0.36%	SKKB	0.36%
NLPC	0.36%	NSPC	0.27%	YKCA	0.18%
BCHRT	0.18%	SKPC	0.18%	ALRB	0.09%
YKSC	0.09%	NWTCA	0.09%	PECA	0.09%
PESCAD	0.09%	NWTTC	0.09%	BCCRT	0.09%
NSSM	0.09%	MBCA	0.09%	YKTC	0.09%
MBQB	0.09%	ABCJ	0.09%	ONSCSM	0.09%
CM	0.09%				

Table 7: Proportion of queries from each court. Courts from human annotators were parsed from the citation however, many were not recoverable due to different formats.

C.2 Province Statistics

Province	Proportion
Ontario	28.01%
British Columbia	22.37%
Alberta	21.62%
Saskatchewan	8.46%
Northwest Territories	5.45%
Newfoundland and Labrador	4.32%
Prince Edward Island	3.01%
Nova Scotia	2.63%
New Brunswick	2.26%
Yukon	1.88%

Table 8: Proportion of queries from each province.

C.3 Query Variations

Variation Type	Description	Example
Change Point of View	Changes perspective of the asker	As a landlord, what can happen if I enter my tenant's apartment without notice?
Next Steps Focus	Refocuses on practical procedure, action steps or finding help	What can I do if my landlord enters my apartment without notice?
Rule Focus	Targets legal rules/principles at play	What determines if a landlord needs to give notice before entering a tenant's apartment
Information Seeking	Expressing request for information/understanding the scenario	Do courts generally require landlords to give notice when entering a tenant's apartment?
Burden of Proof	Focusing on what evidence is needed in the given scenario	What kind of evidence would I need to show entry without notice?
General Interpretations of a Legal Principle	Generalizes from the specific situation to broader interpretations	How do courts interpret the requirement for landlords to give notice before entering their tenant's apartment?

Table 9: All query variations used to vary seed queries in the query generation phase, using the example seed query "What are my rights if my landlord enters my apartment without notice?".

D Additional Results

D.1 Full Results

LLM	Chunk Size	Method	With Rerank	MRR	Recall@10	Recall@25	nDCG@10	nDCG@25
BM25								
BM25	DocLevel	–	✗	0.245	0.193*	0.309*	0.163*	0.208*
BM25	1024	–	✗	0.197	0.113	0.195	0.119	0.147
BM25	4096	–	✗	0.256*	0.152	0.251	0.162	0.197
Hybrid								
Qwen	1024	Hybrid BM25	✗	0.483	0.299	0.454	0.331	0.375
Qwen	4096	Hybrid BM25	✗	0.516	0.320	0.487	0.360	0.405
Qwen	1024	Hybrid BM25 + Kanon2Rerank	✓	0.563*	0.364*	0.548*	0.426*	0.477*
Qwen								
Qwen	1024	–	✗	0.561	0.359	0.520	0.419	0.463
Qwen	2048	–	✗	0.594	0.354	0.524	0.429	0.474
Qwen	4096	–	✗	0.577	0.367	0.555	0.433	0.480
Qwen	8192	–	✗	0.594	0.393	0.582	0.446	0.495
Qwen	1024	–	✓	0.563	0.363	0.548	0.426	0.477
Qwen	2048	–	✓	0.584	0.373	0.571	0.445	0.497
Qwen	4096	–	✓	0.581	0.374	0.588	0.439	0.497
Qwen	8192	–	✓	0.594	0.417*	0.636*	0.456	0.522
Qwen	4096	IterRetGen	✗	0.616	0.387	0.577	0.461	0.509
Qwen	4096	IterRetGen	✓	0.613	0.377	0.577	0.455	0.509
Qwen	8192	IterRetGen	✓	0.632*	0.406	0.601	0.474*	0.527*
Gemma								
Gemma	1024	–	✗	0.602	0.349	0.521	0.422	0.464
Gemma	2048	–	✗	0.570	0.367	0.537	0.417	0.461
Gemma	4096	–	✗	0.583	0.365	0.529	0.419	0.458
Gemma	8192	–	✗	0.574	0.393	0.567	0.426	0.472
Gemma	4096	–	✓	0.584	0.384	0.598	0.443	0.502
Gemma	8192	–	✓	0.598	0.419	**0.643***	0.458	0.527*
Gemma	4096	IterRetGen	✗	0.615	0.393	0.553	0.457	0.488
Gemma	8192	IterRetGen	✗	0.624*	0.429	0.595	0.473*	0.512
Gemma	4096	IterRetGen	✓	0.619	0.391	0.551	0.457	0.489
Gemma	8192	IterRetGen	✓	0.613	0.431*	0.595	0.471	0.510
Gemini1								
Gemini1	1024	–	✗	0.601	0.343	0.505	0.441	0.475
Gemini1	4096	–	✗	0.625	0.378	0.557	0.472	0.507
Gemini1	8192	–	✗	0.639	0.413	0.599*	0.487	0.526
Gemini1	1024	–	✓	0.577	0.372	0.553	0.440	0.490
Gemini1	4096	IterRetGen	✗	0.639	0.389	0.568	0.489	0.526
Gemini1	8192	IterRetGen	✗	**0.655***	0.429*	0.597	**0.508***	0.538*
Gemini2								
Gemini2	4096	–	✗	0.625	**0.449**	0.626	0.489	0.534
Gemini2	8192	–	✗	0.617	0.373	0.521	0.396	0.438
Gemini2	4096	–	✓	0.600	0.422	**0.652***	0.470	**0.538**
Gemini2	8192	–	✓	0.596	0.423	0.564	0.434	0.467
Gemini2	4096	IterRetGen	✗	**0.661***	**0.456***	0.631	**0.521***	**0.560***
Kanon2								
Kanon2	1024	–	✗	0.536	0.329	0.500	0.385	0.431
Kanon2	4096	–	✗	0.546	0.336	0.541	0.394	0.453
Kanon2	8192	–	✗	0.521	0.352	0.560	0.388	0.451
Kanon2	1024	–	✓	0.559	0.365	0.547	0.424	0.473
Kanon2	4096	–	✓	0.573	0.364	0.572	0.429	0.486
Kanon2	8192	–	✓	0.590*	0.406*	0.624*	0.449*	0.514*
Kanon2	4096	IterRetGen	✗	0.577	0.369	0.555	0.431	0.479
Kanon2	8192	IterRetGen	✗	0.564	0.394	0.577	0.439	0.483

Table 10: Retrieval metrics across all experiments. Higher is better for all metrics. Bold Underlined = overall best, Bold = 2nd, Underlined = 3rd. * denotes best within group.

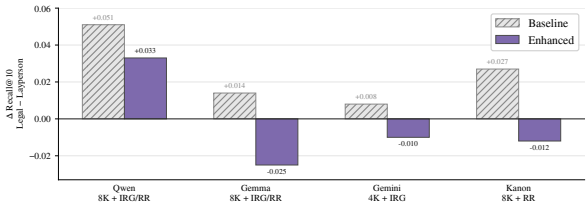

Figure 4: The average difference between Recall@10 of retrieval methods on legal and layperson style queries. The selected configurations are the base dense retrieval and the high performing configurations of the different embedding models. Positive values indicate a the retrieval system had higher performance on legal style queries.

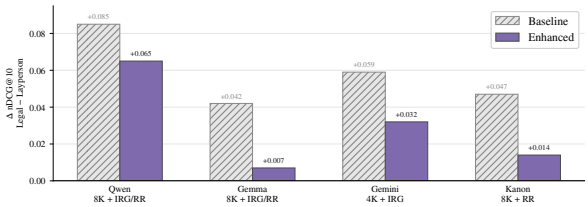

Figure 5: The difference between nDCG@10 of retrieval methods on legal and layperson style queries. The selected configurations are the base dense retrieval and the high performing configurations of the different embedding models. Positive values indicate a the retrieval system had higher performance on legal style queries.

D.2 Analysis of Unsupported Claims (Accuracy)

Of the three models, `Qwen` achieves the highest accuracy by a large margin. With that said, to better understand the type of errors that `Qwen` made, we sampled 25 unsupported claims and manually analyzed them (Table 11). We found that `Qwen`'s answers were significantly longer than the gold standard answers (that had an average of 3883.5 characters), and often contained claims that were either irrelevant, too detailed, or relevant but miss the point of the question (such as talking about factors that impacted other applications for employment insurance when it's already established the client is ineligible). A small number of errors include hallucinations (i.e. claims unsupported by the documented they cite), false positives (claims that appeared in the gold answer), and relevant claims missing from the gold answer.

Error Category	Percentage	Description
Too detailed	24.0%	Contained more details that were not necessary
Irrelevant	24.0%	Discussed irrelevant facts about other issues
Misses the point	32.0%	Answered other questions not helpful to the query
Relevant	4.0%	Actually relevant but missing from the gold answer
Hallucination	8.0%	The statement was not in the answer, nor was it grounded in retrieved documents.
False Positive	8.0%	The statement was judged as not being in the gold answer but was actually

Table 11: Categorization and distribution of 25 Oracle generation statements which were judged as not being entailed by the statements in the ground truth answer.

D.3 Categorization of unsupported claims

Error Category	Percentage	Description	Example
Conversational Penalty	38.46%	The LLM judge penalizes the generator for discussing opening or synthesizing documents in the concluding the answer.	An answer contained "The inquiry concerns the employment insurance eligibility of a part-time employee who voluntarily left after a reduction in hours" which was not a statement in the oracle documents. Another generated claim mentioned having to do a case-by-case basis due to different documents having different outcomes based on different facts. This 'case-by-case' synthesis was judged as ungrounded
Judge Decontextualization Error	3.85%	The decontextualizer confused entities in the claim	The government was swapped for the employer
Hierarchical Flattening	11.54%	The generated answer elevates non-binding text, minority opinions, or specific factual reasoning to authoritative statutory law.	A generated answer confused the minority opinion for the majority opinion binding law in Canada v. Vavilov
Misapplication of Law	11.54%	The generated answer contains misapplications of the law.	Applies incorrect test for litigation privilege to determine solicitor-client privilege because they both appear in the retrieved document
Scope Shifts	7.69%	The generated answer softens a strict law or turns an option into a definite prerequisite.	A retrieved document says 'shall not' definitively but the generated answer says 'may not'.
False Insufficient Evidence	7.69%	The generated answer incorrectly claims the retrieved context lacks the answer.	A generated answer correctly finds the burden of proof in the reasoning (supporting arguments) but in the conclusion states that there is not enough evidence to determine the burden of proof.
Entity Confusion	7.69%	The generated answer confuses entities.	A generated answer said the applicant has a specific responsibility but the retrieved document explicitly says the Commission (government) has that responsibility."
Cites Wrong Document	7.69%	The generated answer states a grounded fact but misattributes it to a different retrieved document which is not related to that fact.	In a generated statement discussing the value of the context of tweets, it cites a document which never discusses tweets or Social Media, while a separate retrieved document extensively discusses the topic.
Confused about Document Formatting	3.85%	The generated answer misattributes the section of a rule due to formatting artifacts in the retrieved chunk.	The section number of a rule appeared in the document header while the section occurred partway through the page, the section number from the header was placed before the actual section.

Table 12: Categorization and distribution of 25 Gemma generation statements which were judged as ungrounded by the oracle retrieved documents. Around 40% are over penalizations while the rest illustrate a variety of hallucinations that LLMs face in the legal domain. Note that one claim was categorized into two error types.

Error Category	Percentage	Description	Example
Conversational Penalty	56.00%	The LLM judge penalizes the generator for discussing opening or synthesizing documents in the concluding the answer.	A generated claim states that a significant reduction in hours is considered when assessing just cause, but the LLM judge flags it as ungrounded because “reduction in hours” is absent from the retrieved document which discusses conditions that significantly modify wages.
Other Judge Error	4.00%	Other LLM judge errors	In one generated answer, one document discussed facts less related to the statement while another document contradicted it using more relevant facts but the LLM judge discussed that the statement was not grounded because the former document contradicted it.
Misapplication of Law	12.00%	The generated answer contains misapplications of the law.	In one generated answer, a very long retrieved document placed at the start of the context explicitly stated that solicitor-client privilege prevents a certain disclosure, however the generated statement cites it and makes the opposite claim, accidentally negating it.
Misattributing Jurisdiction	4.00%	The generated answer cites a document and mistakes its jurisdiction.	In one generated answer, it explicitly cites a BC supreme court case where the court and province is provided and attributes it to Yukon courts to tailor it to the query.
Scope Shifts	4.00%	The generated answer softens a strict law or hardens a permissive statutory option into a mandatory prerequisite.	An example answer inaccurately swaps from "must do so if requested" to "must do so"
Cites Wrong Document	8.00%	The generated answer states a grounded fact but misattributes it to a different retrieved document or an incorrect level of the legislative hierarchy.	An example answer cites the Alberta Human Rights Code instead of Act.
Creates assumption in Query	4.00%	Assumes an important fact based on query statements that do not actually support the claim	In a generated answer, it assumes that the user suffered so much burnout as to cause mental disability despite just saying they took sick leave due to burnout without additional details.
Using External Knowledge	8.00%	The generated answer uses knowledge from pretraining to supplement answer.	In one generated answer, the retrieved document stated "law grants the specific power to prosecute actions on behalf of an estate to the estate’s formal ’executor or administrator’", the answer used external logic of "an affirmative grant of authority to a specific legal role inherently excludes those who do not hold that role." to expand its point.

Table 13: Categorization and distribution of 25 generated statements for the Gemini pipeline on LegalCadRetrieval-Bench. Around 60% of flagged errors are LLM judge over-penalizations due to conversational text or connecting text between documents, while the remaining errors capture genuine structural slips like doctrinal conflations and statutory misattributions.

E Models Used

Model	Type	Task
Qwen3-Embedding-8B	Open	Embedding
EmbeddingGemma	Open	Embedding
Gemini-Embedding-001	Closed	Embedding
Gemini-Embedding-2	Closed	Embedding
Kanon-2-Embedding	Closed	Embedding
Kanon-2-Reranker	Closed	Reranking
Qwen-3.5-9B	Open	Text generation
Gemma-4-E4B	Open	Text generation
Gemini-Flash-2.5	Closed	Text generation
Gemini-Pro-2.5	Closed	Text generation, Judging

Table 14: Evaluated models and categories. All open embedding models run on A6000s for 0.5-2 hours. Open Generator models ran for 8-24 hours.