LEVERAGING LANGUAGE MODELS FOR DOCUMENT SUMMARIZATION IN THE ROMANIAN JUDICIAL SYSTEM

Vlad-Andrei BĂDOIU[1], Gabriel GUTU-ROBU[2], Cosmin STEREA-GROSSU[3], Ovidiu GHIBEA[4], Mihai-Valentin DUMITRU[5], Ciprian DOBRE[6]

Despite the widespread adoption of Large Language Models (LLMs) across industry and research communities, the Romanian public sector has been reluctant to integrate them into its digitalization efforts. This reluctance has many sources including concerns about implementation costs, limited technical expertise, and the risk of model hallucinations. While LLMs do not represent a universal solution, they offer significant potential for streamlining tedious tasks such as document completion, summarization, and data extraction.

This paper explores one particularly labor-intensive but important task: generating summaries of a court file, required in the first part of juridical decisions. We describe the development of an LLM-based solution for this specific task, from model selection and training, system architecture to the implementation of hallucination-reducing techniques. We designed the system not as a replacement for human expertise, but as a tool that can significantly reduce the time spent on this task; in the last part of the paper we discuss integration into the human workflow and how to make it easy to validate the generated summary. Our evaluation shows that LLMs are great candidates for this task, achieving high quality summaries with over 90% semantic similarity with human-written summaries.

Keywords: Large Language Model (LLM), Automated Text Generation, Natural Language Processing (NLP), Semantic Analysis, Public Sector, Artificial Intelligence (AI)

[1] PhD Student, Faculty of Automatic Control and Computer Science, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: vlad_andrei.badoiu@upb.ro

[2]Lecturer, Faculty of Automatic Control and Computer Science, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: gabriel.gutu@upb.ro

[3]Judge Superior Council of Magistracy, Bucharest, Romania, e-mail: cosmin.sterea@csm1909.ro

[4]Masters Student, Faculty of Automatic Control and Computer Science, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: ovidiu.ghibea@upb.ro

[5]PhD Student, Faculty of Automatic Control and Computer Science, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: mihai.dumitru2201@upb.ro

[6]Professor, Faculty of Automatic Control and Computer Science, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: ciprian.dobre@upb.ro

1. Introduction

These days, Large Language Models are at the center of attention due to their remarkable capabilities. Commercial models, now reaching trillion-parameter scales, have shown impressive results in natural language processing, code generation, and even multimodal tasks [19]. They are quickly adopted in industry with new use cases being explored [1]. However, a major challenge in adopting commercial LLMs involves data regulations. In many cases, companies and institutions are not allowed to send sensitive data to external services. This is especially true for the public sector, where data security and confidentiality are critical.

For such use cases, open-weight models provide a viable alternative. They are competitive for specific tasks [20] and can be run on-premises. Alongside a growing research community rapidly developing tools and techniques to reduce memory footprint and increase inference speed, widespread adoption of LLMs is becoming increasingly feasible. Despite this democratization and rapid adoption across industries, the public sector has been notably slower in introducing LLMs in practical applications. This hesitancy stems from several factors: operational costs, hardware requirements for local deployment, expertise gaps, and concerns about model hallucinations.

On the other hand, the legal domain, which intersects closely with the public sector, has shown more progressive adoption of LLMs for various applications [1]. Examining existing market solutions confirms that LLMs can effectively automate many routine legal tasks. We therefore consider the judicial system a prime candidate for LLM adoption in the public sector.

Specifically, we address the tedious task of summarizing case files. In the Romanian judicial system, court decisions must include an abstractive summary of all relevant documents. Currently, this task is performed manually, which can be tedious, especially for cases with numerous or lengthy documents. To this end, we built an LLM-based solution designed to be used in a human-in-the-loop fashion, where the generation process is broken into smaller steps that users can quickly validate.

To achieve this, we explore two approaches: (i) a judge-based approach where several summarizations are generated directly from all the documents and a judge model selects the best one, and (ii) a hierarchical approach where we first extract the relevant entities such as the legal argument, parties involved, etc., and we process each extracted entity separately and compose the final summary algorithmically. In this work we make the following key contributions:

- We design and implement an open-source LLM-based system for Romanian court case summarization, exploring two distinct architectures: a judge-based system suitable for scenarios with minimal training data, and

[1] https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai-2024

a hierarchical system that achieves human-level performance through task decomposition and specialized LoRA adapters.
- We show that 8B parameter open-weight models can achieve excellent results in specialized legal domains when properly fine-tuned and integrated
- We present a comprehensive methodology for building production-ready legal AI systems, including OCR preprocessing for Romanian documents, entity verification mechanisms to reduce hallucinations, and human-in-the-loop design patterns that enable efficient validation of generated content.

2. Related Work

The use of LLMs in the legal domain has been explored in both industry and research contexts. Commercial applications include software to assist the lawyers with data analysis, identifying relevant case law, statutes, and legal precedents [1].

From a research perspective, LLMs in the legal domain have been employed to assist with various tasks including providing legal advice to users [15], assisting judges during trials, extracting key elements from legal documents, automatically generating legal content, matching optimal solutions for cases, performing case logic reasoning, and simplifying judicial procedures to improve efficiency [1]. Legal judgment prediction systems employ both general and specialized architectures. Dataset development includes the ECHR corpus for multi-label classification [4], while models like AutoLAW learn from legal argument-precedent pairs [21] and PekoNet integrates summarization into prediction for colloquial descriptions [22]. Domain-adapted models like jurBERT demonstrate the effectiveness of specialized pre-training for legal NLP tasks [23].

A key challenge in the legal domain is the need for high accuracy and reliable summaries. There are two types of summaries: extractive and abstractive. Extractive summaries are based on extracted text from the original text, whereas abstractive methods introduce new text and wordings that are not present in the original text. Legal document summarization approaches encompass general methods and domain-specific solutions. Traditional supervised and unsupervised techniques have been applied with extractive and abstractive approaches [26], while specialized methods like LegalSumm leverage textual entailment for case ruling summarization [24] and DELSumm incorporates legal expert guidelines [25]. For extractive methods, language models show great results, judged to be on par with human written summaries [1]. However, for more complex scenarios where abstractive summarization is needed, LLMs can produce inconsistencies [2, 3] or even summaries that are factually incorrect [27]

LLM adaptations for summarization include iterative frameworks like SummIt that address hallucination through self-evaluation [13] and element-aware methods that enumerate key facts [14]. Advanced techniques include SliSum's sliding window approach for long documents [12] and DiffuSum's diffusion-based extractive method [11].

3. Dataset

At the heart of our systems is a specialized LLM for the entity extraction and summarization task of Romanian legal documents. To train the model we leveraged case documents from the ReJust platform administrated by the Superior Council of Magistracy (SCM), which is a system used by courts in Romania for managing court cases.

The training dataset consists of court files, which have specific structure determined by the court type or legal procedure. They usually have an associated national number assigned by the courts' system and follow a standardized format that includes relevant information about the case file. The format is XXXXX/II/YYYY, where:

- XXXXX is a unique sequential number assigned to each case file opened in that year by a specific court;
- II is a fixed code assigned to the court (usually a number or a prefix representing the court or type of proceeding);
- YYYY is the year in which the case file was opened.

A legal case file is composed of several types of documents that are used to draft the summary. Generally, these documents fall into a relatively narrow range of categories, with each category having its specific entities and structure. The human-written summary is part of the court decision document. It should be noted that in the current implementation of the ReJust web system, the user has to mark the documents relevant for the summarization. This is because there are situations where the file includes additional documents (such as annexes or invoices) that are irrelevant to drafting the summary but are still part of the file, being used for other purposes.

3.1. Dataset Characteristics

At this stage, we have identified 23 843 court files which contain the court decision and the relevant summary. We note that some of them may not be usable due to missing documents. Of these, approximately 58% (13 809 files) contain at least two documents relevant to the decision. Each file has some human annotated specific entities: the legal parties, the legal arguments, the claims, the legal grounds, and the request. Depending on the file type, they have slight variations, for example in the statement of defence, the defendant's arguments are presented instead of the plaintiff's arguments.

A statistic on the most frequent documents found as part of a legal case file is available in Table 1. It can be noticed that the Statement of Claim

is the primary document forming the basis of a case file. The next most common document is the Statement of Defense, present in about 70% of the files. The Appeal Request is the following document, found in 10% of the files. Subsequently, the frequency of appearance of other documents decreases, but this does not indicate the exclusion of the corresponding file, as these documents are generally found together with other documents.

TABLE 1. Types of documents found in at least 1% of the case files.

Document Name	***Number of Case Files***	***% of Case Files***
Statement of Claim	23 843	100%
Statement of Defence	16 587	70%
Appeal Request	2 482	10%
Claim	2 337	9,8%
Reply to the Statement of Defence	1 317	5,5%
Written Notes	1 132	4,7%
Clarifications	1 086	4,6%
Objection to Enforcement	741	3,1%
Closing Arguments	479	2%
Application for the Authorization of Forced Execution	469	2%
Notice of Appeal	298	1,2%
Reasons for Appeal	276	1,2%
Request for Re-examination	261	1,1%

3.2. **The Structure of a Court File**

The provision of details for a court file is done through an endpoint that takes a national case file number and returns a JSON with an array of processing results for the case file by one or more courts. For each court involved in resolving a case file, we have the following information:

- Metadata and specific attributes of the case file (e.g., internal identifiers, case file date, name of the legal matter, name of the court, and others)
- List of involved parties
- List of hearings
- List of documents

3.3. Text Extraction from Documents

This data is provided in formats (e.g. PDF) which include text, scanned documents, or even photographed documents. While the platform itself provides OCR functionality, we opted to extract the text ourselves. This is because we wanted to remove the footnotes and other irrelevant information and bypass scanned pages with no relevant text for the summarization task. To extract the text for the training dataset, we use the Tesseract OCR engine [9], as it has support for the Romanian language and its diacritics, which other OCR engines do not. To improve the quality of the extracted text, we applied several techniques.

Our approach works by processing one page at a time and does not preserve the continuation of multi-page footnotes, for example. Using the `pypdfium2`[2] framework, we extracted text bounding-box information. These bounding boxes don't always form an outline around an intuitive part of text, such as a line, a paragraph, or a visual block. For example, sometimes text containing the special Romanian alphabet characters (ă, â, î, ș, ț) had these in their own boxes, with text before and after them boxed together. To address these issues, we clustered the bounding boxes with the DBSCAN [10] algorithm, using the smallest possible distance between any two points on the perimeter of two boxes as the distance metric. The epsilon parameter, which is the maximum distance between two points in the DBSCAN clustering algorithm (in our case, text bounding boxes) for them to be considered part of the same cluster, was heuristically chosen to avoid clustering together different text elements (such as paragraphs and image captions), but also compensate for large spacing, ensuring lines from a paragraph are clustered together.

We then iterate through the resulting clusters, and compute visual lines: lists of bounding boxes that are visually at the same height. Our goal is to return a list of strings, where each string should ideally be a self-contained unit, such as a sentence. To this end, we iterate through all the clusters and the lines within, heuristically eliminate clusters and lines with low-quality text, then attempt to build sentences by merging visual lines and breaking them where punctuation appears.

A common problem with poorly-encoded Romanian texts, is the substitution of some letters ("ã" instead of "ă", "º" instead of "ș"), because of non-UTF-8 encodings. We apply a dictionary-based fix to address the well-known pairs.

4. Model Selection and Training

Currently, there are several open-weights models that have been trained on Romanian that we identified as candidates due to the model sizes they offer: Llama[5], Qwen [17], Gemma[6] and Ministral[3]. We evaluated these

[2] https://github.com/pypdfium2-team/pypdfium2
[3] https://mistral.ai/news/ministraux

models based on their performance in summarization tasks, specifically using ROUGE-1 and ROUGE-2 scores between the generated text and the human-written ground truth. We provided each model with the full text of the court case documents as input and asked it to generate a summary.

We used the models variants with 8B and 12B parameters, as these are the most suitable for our use case. We need to be able to scale for thousands court cases per week, work with large context windows, while also being able to run on a one node GPU server.

The results are shown in Table 2, where the recall is reported. We note that ROUGE score is not a perfect metric, as it measures the overlap of n-grams, and the juridical language is very specific. The base models do not have the specific knowledge and language constructs needed for the legal domain, and thus the scores are relatively low for ROUGE as they use a completely different language.

We can see that the models perform similarly, with Gemma 3 12B slightly outperforming due to its larger size. Combined with a qualitative analysis of the generated text, we observed that Llama 3.1 8B tends to hallucinate less and produce similar quality text to Gemma 3 12B. Given the lower parameter count and the context size of 128k tokens, we selected Llama 3.1 8B as the base model for our system.

TABLE 2. Model Performance Metrics

Model	**ROUGE-1**	**ROUGE-2**
Gemma 3 12B	0.3287	0.1398
Llama 3.1 8B	0.2730	0.1124
Qwen3 8B	0.2640	0.0865
Ministral 8B	0.2187	0.0846

With the base model selected, we proceed to specialize the model for the summarization task. We considered two methods: Few-shot learning and fine-tuning. Few short learning is a technique that allows the model to learn from a few examples introduced in the prompt, thus improving the accuracy and reliability of text generation. A problem we encountered is that the size of the multiple court cases can exceed the context windows of 128k tokens. This is caused by the low compression rate of the tokenizer for the Romanian language and the high average number of tokens per court case, around 60k tokens. Thus few-shot learning is not a viable solution.

The second approach is to fine-tune the model. We considered two techniques: continuous pretraining [7], which involves continuing the model's learning phase with a new dataset. However, the costs are far higher than a public institution can afford, with tens of data center grade GPUs being needed just to hold a copy of the model. To this end, we employed LoRA adapters[8], where

the model is frozen during the training phase, and a new block of parameters is introduced and subjected to the training process.

5. Building a System for Document Summarization

We designed our system as an assistant to human operators, providing the necessary means for quick validation of generated text and keeping hallucinations to a minimum. We have explored two architectures for this task: a judge based system and a hierarchical summarization approach. The former works in more constrained scenarios where no annotated data is available, but is less flexible. The latter requires annotated data but is more fine-grained, enabling easier validation and better final results.

The entire system is written in 4000+ LoC in Python and released as an open-source project[4]. The system is deployed using Docker Compose and uses FastAPI for the exposed REST API. Tasks are orchestrated using Celery, with Redis as a message broker. In the remainder of this section, we describe the two architectures in detail.

5.1. Judge Based System

The Judge based system is presented in Fig. 1. This is the first iteration of our system, and is designed to work with minimal training data: the pair (documents, summary).

The user inputs all the court case documents in PDF format into the system (①). At ②, the OCR module takes the input documents and returns, for each document, a JSON file with the extracted text. The JSON files are then passed to the model for entity recognition with the base LLM (③) and to the LoRA-adapted model for summarization (④). The summarization model is used to generate k candidate summaries.

Next, to assess the correctness of the generated text, we need to identify the legal entities referred to in the generated text (articles, names, and other legal entities) and ensure that no new entities have been introduced by the model (⑤). To this end, we take a two-step approach: a simple rule-based method and one that leverages LLMs.

We first identify all the law articles and replace names with name tags using predefined regex rules. Next, if no new entities have been introduced, we ask an LLM to identify other missed entities. To verify the entities returned by the LLM, we ask it to provide regex rules to match the entities. As we are operating in a constrained environment, we use the base model without the LoRA adapter. Next, we perform a similar search in the generated text. If we identify new entities, we drop the summarization candidate; if only 1 to 5 have been introduced, we drop the sentences containing them. Furthermore, we drop the summaries that have less than 10% overlap in entities with the source documents.

[4] https://github.com/upb-nlp/juridoc

Finally, the summaries passing the entity check are passed to a judge model to grade them. We use N judges to give a score for each summary. The highest scoring summary is selected as the final result (⑥).

To reduce the validation overhead incurred by the user, the system returns a JSON with the generated summary alongside references to the source paragraphs that were used to build the summary. This approach reduces validation time; through the UI over the API, this enables side-by-side visualization of the summary and clickable paragraphs that link to the source text.

The source paragraphs are identified by computing the cosine similarity between each paragraph from the generated text and all paragraphs in the source documents. The paragraphs that pass a threshold of 0.5 are considered relevant. This is an iterative process: the user can select or deselect paragraphs in the source documents and recreate a refined summary (⑦), thus establishing a feedback loop to quickly correct the summary.

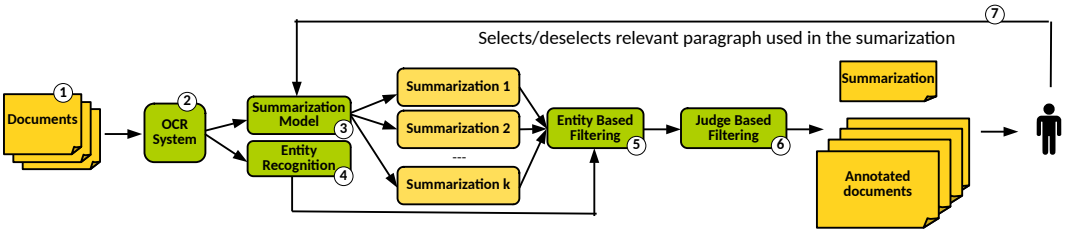

FIG. 1. Judge based system architecture.

5.2. **Hierarchical Summarization System**

The hierarchical summarization architecture takes a two-step approach as shown in Fig. 2: first, we extract relevant entities from each document type (e.g., Statement of Claim), and then we process each extracted entity separately, composing a final summary programmatically from the processed entities. While we require annotated data for each document type to train the entity extractors and processors, this approach offers more granular control and better validation opportunities.

5.2.1. *Entity Extraction.* We first extract relevant entities from each document type. There are two main document types that we target, which cover most court cases: Statement of Claim and Statement of Defence. For these, we extract five key entities: (1) the parties involved, (2) the legal grounds, (3) the claims, (4) the legal arguments, and (5) the evidence. Most other document types can be mapped to one of these five entities and reuse the same extraction and processing models.

For each entity type, we employ a specialized LoRA adapter trained to annotate the original document. The system receives the OCR-extracted text

of a document and determines, for each word in the text, whether it is part of one of the entities or not. By using this approach, we have no hallucination issues, as we only tag the words in the original text. This is similar to how the existing ReJust platform works, where users can highlight relevant text spans from the interface for each of the five entities. Our approach fully automates this process, and users can visually validate the extracted entities, as the text spans are highlighted in the user interface as they would normally appear in ReJust when done manually by a human operator. Once users validate the extracted entities, the annotated document is passed to the next step for processing.

5.2.2. *Entity Processing.* After validation, we process each extracted entity type using specialized LoRA adapters. The processing task varies depending on the entity type:

- Legal arguments: generate a concise summary of the legal arguments presented in the document, ensuring clarity and coherence.
- Claims: extract and standardize the format of claims made by each party.
- Legal grounds: extract and standardize the legal grounds cited in the document.
- Parties involved: extract the names of the parties involved, their roles, and gender information to be used later for pronoun agreement.
- Evidence: extract the evidence presented and rewrite it in a standard format.

During development, we observed that extracted entities often lack the necessary context to determine correct pronoun usage and word inflection, including gender and number agreement. For example, in the claims section, the model would use incorrectly write "a solicitat" (*has solicited*) instead of "au solicitat" (*have solicited*) when there were multiple plaintiffs. To address this issue, we use a dedicated LoRA adapter to extract gender information from the text about the parties. For number agreement, we assume plural if there are multiple parties, otherwise singular. The user can validate these attributes before proceeding to entity processing.

This grammatical information is used to adjust the prompts when processing each entity, ensuring that the generated text maintains coherence and uses correct pronouns throughout. As we did not have annotated data for processing, we leveraged a much larger model, the Llama 3.1 405B-parameter model, prompted with few-shot examples to generate synthetic data for training the LoRA adapters.

With the entities processed, the user platform of our system can compose the final summary based on its own rules.

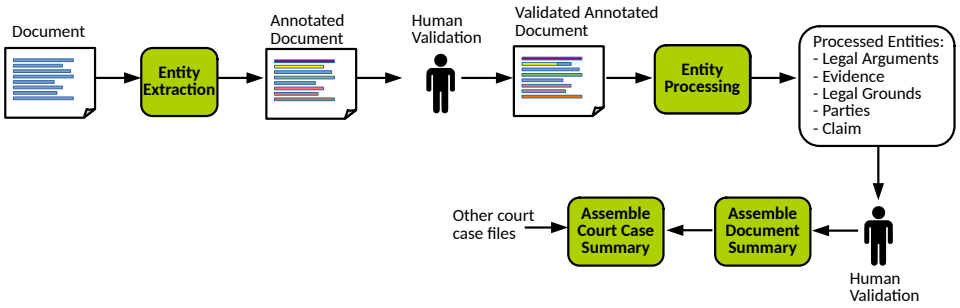

FIG. 2. Hierarchical system architecture.

TABLE 3. Training Parameters

Parameter	Value
LoRA rank (R)	32
Learning rate	10^{-4} (0.0001)
Training epochs	4
LR scheduler type	cosine

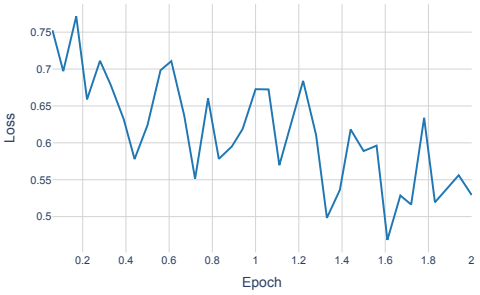

FIG. 3. Loss during training.

6. Evaluation

We evaluate our system from two perspectives: (i) accuracy for the court file summarization task and (ii) the performance of the system in terms of number of requests that can be processed per GPU.

For the training software stack we leveraged LlamaFactory [29] with LongLora [28], as it provides a simple interface and supports most models out of the box. We trained with LoRA [8] on a DGX with 8 A100 GPUs of 80GB VRAM each, using FlashAttention [18].

For the judge based system, where we use the (documents, summary) training pair, from an initial batch of 2000 court cases, we filtered badly OCR-ed documents, and court cases from which we could not automatically extract the human-written summarization automatically; this resulted in a training dataset of 700 samples. The relevant training parameters are shown in Table 3, determined after several iterations of ablation studies. The LoRA rank was set to 32.

During the initial training process we identified several loss spikes, caused by badly OCR-ed documents, for which we removed the badly OCR-ed documents and continued the training from a previous checkpoint. The final train

loss is shown in Fig. 3, with no indication of overfitting, further confirmed during the evaluation phase.

In a similar manner, for the hierarchical summarization system, we trained 10 LoRA adaptors for entity extraction: 5 for the Statement of Claim and 5 for the Statement of Defence. We only used 2 training epochs for them. We used for training a dataset of 14589 claims and 2752 defences. For the entity processing models, we used a synthetic dataset of 4000 samples generated with the Llama 3.1 405B model, for each type of entity. We validated all the samples manually and used several heuristics for a first phase cleanup to remove badly generated samples:

- we checked that the generated text has the verbs at the right tense, by using a dictionary of some common Romanian verbs and their conjugations
- for rewriting tasks we removed samples that had a word count of less than 50% or more than 50% of the original text as rewriting should not change the length too much
- for summary asks, we removed the samples that either introduced new entities or dropped too many entities from the original text

This has resulted in another set of 10 LoRA adaptors. For all other document types we reuse the existing LoRA adaptors.

6.1. **Entity Extraction**

To measure the performance of the entity extraction models, we evaluated the performance on 400 samples that were not part of the training set. We measured the recall and precision and plot the combined results for all document types in Fig. 4.

The results show that the LoRA-adapted model is excellent at extracting the entities, with a recall of 84% for the largest entity by the number of words, the legal arguments. While for the claimants we achieve lower precision, we explain this as the way the data is annotated: in many cases, the annotators highlight more text than necessary, such as the name of the claimant and the address, while we only need the name. Overall, the results are very good, and the user has only to make minor adjustments before sending the entities for processing.

6.2. **Summarization Accuracy**

As there are no existing models or benchmarks for the Romanian language and court case format, we valuated our system on a set of 100 diverse court cases, which were not part of the training dataset. We selected the cases to have different types and a distribution similar with the court cases statistics we have presented in the dataset section.

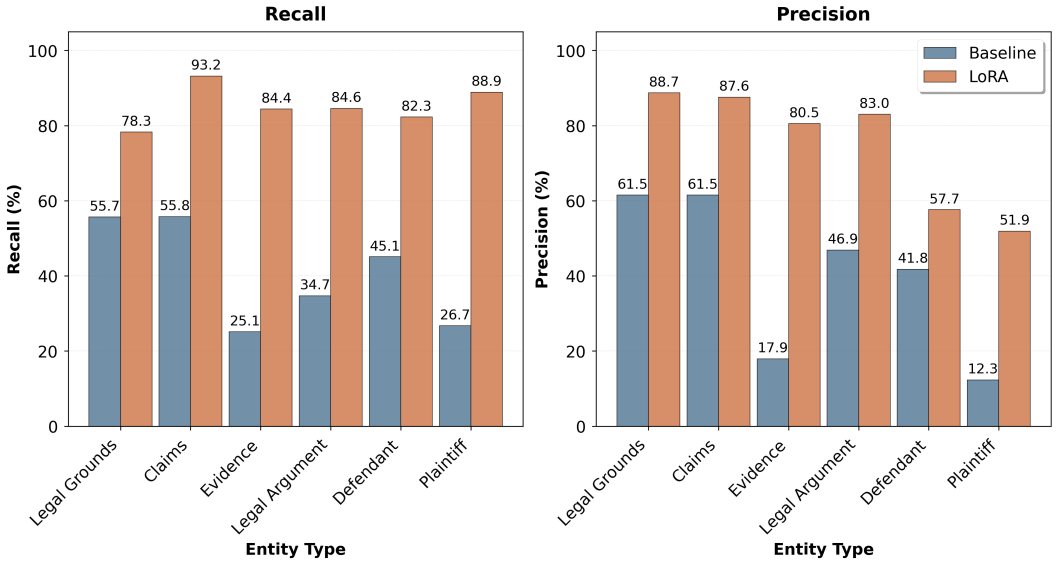

FIG. 4. Entity extraction performance comparison between baseline model and LoRA-adapted model across different entity types.

We compared the system's output against human-produced summarization using multiple evaluation metrics. ROUGE-1 measures unigram overlap, capturing the presence of individual words from the reference summary. ROUGE-2 evaluates bigram overlap, reflecting phrase-level similarity. ROUGE-L measures the longest common subsequence, indicating structural coherence between the generated and reference texts. Additionally, we employed BERTScore [16] to assess semantic similarity, which is particularly important for legal texts.

Table 4 presents the performance of different system configurations, showing the progressive improvements achieved through architectural enhancements. Each row represents an incremental step in our development process, from the baseline model to the judge based system and the final hierarchical system.

Baseline Performance. The base Llama 3.1 8B Instruct model, despite few-shot prompting and detailed instructions, produces inadequate summaries with limited legal terminology and generic language that fails to capture the specialized discourse of court decisions. The low BERTScore (0.641) confirms poor semantic alignment with human-written summaries.

Judge System Performance. The judge-based system demonstrates progressive improvements through three architectural enhancements. The initial LoRA adapter significantly improves results, with BERTScore increasing to 0.754. Adding summary aggregation and the full system (including entity recognition and judge model) further boosts performance to a BERTScore of

TABLE 4. Summarization performance evaluation on 100 court cases. Judge System (Full) includes LoRA adaptor, aggregator, entity recognition, and judge model. Higher scores indicate better performance.

System Configuration	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
Llama 3.1 8B (baseline)	0.264	0.111	0.149	0.641
Judge System (Adaptor)	0.337	0.206	0.223	0.754
Judge System (Adaptor + Aggregator)	0.368	0.209	0.250	0.772
Judge System (Full)	**0.390**	**0.249**	**0.288**	**0.820**
Hierarchical System	**0.628**	**0.411**	**0.510**	**0.904**

0.820 and ROUGE-L of 0.288.The resulting summaries provide a high semantic similarity with the human-written summaries, as well as a good overlap of ROUGE scores. The generated summaries show less inconsistencies in the entities and better overall structure. The model is able to use the legal terminology correctly. The model is able to generalize and improve based only on the data in the form of the pair (documents, summary), showing promising results even in this constrained setting. However, a qualitative analysis reveals several issues:

- The model tends to end responses by repeating information, a common issue with smaller LLMs.
- Inconsistencies appear in extracted entities, particularly dates.
- Confusion between plaintiff, defendant, and other party designations.
- Minor procedural details emphasized over main case facts.
- Events presented out of chronological order.

Hierarchical System Performance. The hierarchical summarization system shown in the last row of Table 4 shows a significant improvement over the judge-based system. By breaking down the summarization task into smaller sub-tasks, we are able to produce human-level summaries, with a BERTScore of 0.90. Looking at the generated summaries, we observe that they fully follow the standard structure of the summary, use the correct legal terminology, are factually accurate, and correctly summarize the main events of the case in a coherent manner. This shows that with a carefully engineered system and well-annotated data, small LLMs can achieve excellent results in specialized domains. This system is now in production use by several courts

in Romania, from which we have received positive feedback regarding the usefulness of the system and the overall quality of the generated summaries.

6.3. **Performance**

Performance is a critical aspect of our system, as it needs to be able to scale easily for multiple courts. To this end, we leverage PagedAttention via vLLM [30], which reduces the overall memory usage and increases the number of requests per second. A second important aspect in choosing vLLM is the ability to quickly switch between LoRA adapters. We set the maximum GPU utilization to 70%. We evaluated the performance of our system on a setup with VMs in OpenStack with access to an A100 40GB PCIe, an older generation of NVIDIA GPUs.

For the judge-based system, to optimize the throughput, when we generate multiple summaries for the same request, we run 3 of them in parallel, as they share the same prefix. We run a maximum of 3 requests in parallel because a single request can use up to 30% of the KV cache. We were able to achieve a throughput of 300 output tokens per second, which translated to 4 minutes per court case, including the OCR processing time for a system with 10 base translations.

The hierarchical summarization system, while it has extra overhead incurred by the higher amount of context switching between the different LoRA adapters, ends up processing much less data during the processing of entities for the final summary, as we only process the extracted entities, not the entire document. This can be balanced, with the annotation being done offline, at night, while the entity processing and final summary composition are done on-demand. Our evaluation shows a processing time of around 2 minutes per court case. Furthermore, as we're not processing all the court case documents at once, we are able to run up to 20 requests in parallel.

7. **Discussion and Future Work**

In this work we presented a comprehensive approach to building an LLM-based assistant system for summarizing Romanian court case files. We have shown that language models with lower parameter count can achieve excellent results when properly fine-tuned and integrated into a human-in-the-loop framework. With a fine grained approach, where the task is decomposed into smaller steps that are easier to validate, the system is less prone to hallucinations and reduces the work of the human operator considerably.

Our results show that effective court case summarization is within reach with a relatively modest amount of hardware and training data, provided that the system is carefully engineered and tailored to the domain. The current approach, which leverages open-weight LLMs with LoRA adapters and a robust human-in-the-loop validation pipeline, yields summaries that are both semantically faithful and practically useful for human operators.

The system has several limitations that we plan to address in future work. Foremost, the training and evaluation data used in this work is confidential and cannot be shared with the research community, limiting the reproducibility and scope of this work. Secondly, we have observed a bottleneck in the performance of the system in the hierarchical architecture. Performance of the system is limited by the high number of LoRA adapters, which limits the concurrent number of requests, as the current vLLM doesn't run multiple adapters at the same time. To address these limitations and more, where we plan to extend this work in the future in the following directions:

Synthetic Data. Right now, both the the training and evaluation data is confidential and cannot be shared with the research community. To this end, we plan to look into methods to generate synthetic data that mimics the structure and content of real courts, with both OCR errors and realistic legal language. This synthetic data can be publicly shared and would enable further research in this area.

Extended Training and Evaluation Data. While we were able to achieve good results, especially with the hierarchical approach, having more training data would further improve the model's performance and robustness. We plan to work together with the Superior Council of Magistracy to improve the system and expand its coverage to more document types.

8. Conclusion

In this paper, we present an LLM-based solution designed to assist human operators in summarizing court case files within the Romanian judicial system. Our work demonstrates that open-weight 8B parameter models, when properly fine-tuned and integrated into well-engineered systems, can achieve human-level performance in specialized legal domains, being useful in the constrained environment of the public sector.

We explore two distinct architectures tailored to different operational scenarios. The judge-based system is designed for constrained environments with minimal annotated data, where we only have the reference summaries and the original documents. The hierarchical system, our main contribution, decomposes the complex summarization task into sub-tasks with dedicated LoRA adapters, enabling granular human validation at each step. This architecture achieves over 90% semantic similarity with human-written summaries. Beyond the technical contributions, our system has been successfully deployed in production across several Romanian courts, where it has received positive feedback and is being expanded further.

Acknowledgment

This first and fifth authors are funded through the "Romanian Hub for Artificial Intelligence - HRIA", Smart Growth, Digitization and Financial Instruments Program, 2021-2027, MySMIS no. 351416

REFERENCES

[1] *J. Lai, W. Gan, J. Wu, Z. Qi and P. S. Yu*, Large language models in law: A survey, AI Open, Elsevier, 2024.

[2] *Z. Luo, Q. Xie and S. Ananiadou*, Factual consistency evaluation of summarization in the Era of large language models, Expert Systems with Applications, **254**(2024), 124456.

[3] *H. Zhang, P. S. Yu and J. Zhang*, A systematic survey of text summarization: From statistical methods to large language models, ACM Computing Surveys, ACM New York, NY, 2024.

[4] *I. Chalkidis, I. Androutsopoulos and N. Aletras*, Neural Legal Judgment Prediction in English, Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 4317-4323, 2019.

[5] *H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar and others*, Llama: Open and efficient foundation language models, arXiv preprint arXiv:2302.13971, 2023.

[6] *Gemma Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love and others*, Gemma: Open models based on gemini research and technology, arXiv preprint arXiv:2403.08295, 2024.

[7] *X. Jin, D. Zhang, H. Zhu, W. Xiao, S.-W. Li, X. Wei, A. Arnold and X. Ren*, Lifelong pretraining: Continually adapting language models to emerging corpora, arXiv preprint arXiv:2110.08534, 2021.

[8] *E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen and others*, Lora: Low-rank adaptation of large language models, ICLR, **1**(2022), No. 2, 3.

[9] *R. Smith*, An overview of the Tesseract OCR engine, Ninth international conference on document analysis and recognition (ICDAR 2007), **2**(2007), 629-633.

[10] *E. Schubert, J. Sander, M. Ester, H. P. Kriegel and X. Xu*, DBSCAN revisited, revisited: why and how you should (still) use DBSCAN, ACM Transactions on Database Systems (TODS), **42**(2017), No. 3, 1-21.

[11] *H. Zhang, X. Liu and J. Zhang*, Diffusum: Generation enhanced extractive summarization with diffusion, arXiv preprint arXiv:2305.01735, 2023.

[12] *T. Li, Z. Li and Y. Zhang*, Improving faithfulness of large language models in summarization via sliding generation and self-consistency, arXiv preprint arXiv:2407.21443, 2024.

[13] *H. Zhang, X. Liu and J. Zhang*, SummIt: Iterative Text Summarization via ChatGPT, Findings of the Association for Computational Linguistics: EMNLP 2023, 10644-10657, 2023.

[14] *Y. Wang, Z. Zhang and R. Wang*, Element-aware Summarization with Large Language Models: Expert-aligned Evaluation and Chain-of-Thought Method, Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 8640-8665, 2023.

[15] *J. Cui, Z. Li, Y. Yan, B. Chen and L. Yuan*, Chatlaw: Open-source legal large language model with integrated external knowledge bases, CoRR, 2023.

[16] *T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger and Y. Artzi*, BERTScore: Evaluating Text Generation with BERT, International Conference on Learning Representations, 2020.

[17] *A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv and others*, Qwen3 technical report, arXiv preprint arXiv:2505.09388, 2025.

[18] *T. Dao*, FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning, The Twelfth International Conference on Learning Representations, 2024.

[19] *W.-L. Chiang, L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, B. Zhu, H. Zhang, M. Jordan, J. E. Gonzalez and others*, Chatbot arena: An open platform for evaluating llms by human preference, Forty-first International Conference on Machine Learning, 2024.

[20] *S. Subramanian, V. Elango and M. Gungor*, Small language models (slms) can still pack a punch: A survey, arXiv preprint arXiv:2501.05465, 2025.

[21] *R. Z. Mahari*, Autolaw: Augmented legal reasoning through legal precedent prediction, arXiv preprint arXiv:2106.16034, 2021.

[22] *Y.-X. Hong and C.-H. Chang*, Improving colloquial case legal judgment prediction via abstractive text summarization, Computer Law & Security Review, **51**(2023), 105863.

[23] *M. Masala, R. C. A. Iacob, A. S. Uban, M. Cidota, H. Velicu, T. Rebedea and M. Popescu*, jurbert: A romanian bert model for legal judgement prediction, Proceedings of the Natural Legal Language Processing Workshop 2021, 86-94, 2021.

[24] *D. L. Freire, A. M. G. de Almeida, M. de S. Dias, A. Rivolli, F. S. F. Pereira, G. A. de Godoi and A. C. P. L. F. de Carvalho*, Legalsum: Towards tool for evaluation for extractive summarization of brazilian lawsuits, International Conference on Information Technology & Systems, Springer, 258-267, 2024.

[25] *P. Bhattacharya, S. Poddar, K. Rudra, K. Ghosh and S. Ghosh*, Incorporating domain knowledge for extractive summarization of legal case documents, Proceedings of the eighteenth international conference on artificial intelligence and law, 22-31, 2021.

[26] *A. Shukla, P. Bhattacharya, S. Poddar, R. Mukherjee, K. Ghosh, P. Goyal and S. Ghosh*, Legal Case Document Summarization: Extractive and Abstractive Methods and their Evaluation, Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 1048-1064, 2022.

[27] *A. Deroy, K. Ghosh and S. Ghosh*, Applicability of large language models and generative models for legal case judgement summarization, Artificial Intelligence and Law, Springer, 1-44, 2024.

[28] *Y. Chen, S. Qian, H. Tang, X. Lai, Z. Liu, S. Han and J. Jia*, LongLoRA: Efficient Fine-tuning of Long-Context Large Language Models, The Twelfth International Conference on Learning Representations.

[29] *Y. Zheng, R. Zhang, J. Zhang, YeYanhan, Z. Luo*, LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models, Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), 400-410, 2024.

[30] *W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang and I. Stoica*, Efficient memory management for large language model serving with pagedattention, Proceedings of the 29th symposium on operating systems principles, 611-626, 2023.