Indian Legal Text Question Answering and Summarization Using Reinforcement Learning

Pranav Moothedath*, Abhishek Srinivas†, Shreesha M‡, Arnav Santosh§
Department of Artificial Intelligence, NITK Surathkal
*pranavmoothedath.221ai030@nitk.edu.in, †abhisheksrinivas.221ai004@nitk.edu.in, ‡shreesham.221ai037@nitk.edu.in,
§arnavsantosh.221ai010@nitk.edu.in

***Abstract*—Legal documents in India are often lengthy, complex, and difficult for the general public to understand. To make legal information more accessible, we explore a system that can both summarize legal texts and answer questions related to them. In this work, we focus on two independent tasks: summarization and question answering using Indian legal text documents. For the summarization part of our work, we use reinforcement learning to train a model that can generate accurate and meaningful summaries of Indian Legal texts. Reinforcement learning helps the model improve over time by incentivizing and rewarding outputs that are more relevant, concise, and close to expert-written summaries. This approach allows the model to learn from its mistakes and gradually generate better summaries that retain important legal points. For the question answering task, we leverage the Pinecone Database and Groq's LLaMA3 language model. Evaluation of summaries produced is done using ROUGE, and NLI metrics. This demonstrates the effectiveness of our approach in producing accurate and context-aware legal summaries. The question answering system also shows strong performance in producing clear and concise answers for lengthy legal documents. Our system would play a crucial role in the understanding of complex legal texts and would make legal knowledge more accessible to all.**

***Index Terms*—Reinforcement Learning, Pinecone, LLaMA3, ROUGE, NLI**

I. Introduction

The Indian legal system produces a large number of complex documents that cannot be understood by most people. Legal texts (especially court judgements and laws) use difficult language, technical jargon and extremely long and complicated sentences. This creates a huge problem as ordinary citizens struggle to understand the legal information that affects their lives. Moreover, in recent times, there is a trend of court cases being digitized, and thus there is an increasing demand for automated systems that can understand, summarize and answer questions pertaining to legal texts.

Traditional NLP techniques tend to fall short when applied to legal corpora due to their inability to handle longer contexts. They also do not consider domain specific sentiments. To tackle these challenges, we develop a system that can do 2 independent tasks : summarize lengthy legal documents and answer specific questions about them.

For the summarization part, we use abstractive summarization, where the model tries to generate a summary in its own words, similar to how a human would. This is different from extractive summarization, which only picks and copies a few sentences or phrases from the original text. Extractive summaries often miss the bigger picture or fail to connect ideas clearly, especially in legal documents where key points might be spread out across different sections. Also, legal language can be very indirect, so just picking sentences may lead to confusing summaries. We employ reinforcement learning (RL) techniques that allow our model to iteratively and continuously improve its ability to generate concise and meaningful summaries of long legal documents. Our RL approach rewards summaries that stay faithful to the original text and that follow logically from the original text. Our dataset for summarization comprises 7030 training samples and 1000 testing samples of Indian legal documents. We assess our summarization system using multiple metrics including ROUGE and NLI (Natural Language Inference) scores to ensure the summaries are semantically consistent.

For the question answering component, we implement a retrieval-augmented question answering (QA) system that uses the Pinecone vector database for retrieving relevant chunks of the legal text in an efficient manner. After retrieval, we use Groq's LLaMA 3 API for generating natural language answers. The dataset used for testing the efficiency of our RAG implementation consisted of 1897 question answer samples taken from around 300 court cases.

Thus, our contributions in this work include:

- Designing a summarization system that can read long Indian legal documents and generate shorter, meaningful summaries in its own words. We evaluate the generated summaries using ROUGE and NLI scores to ensure their quality and semantic consistency.
- Building a question answering system that helps

users ask questions about legal documents and get clear, relevant answers. This is achieved using Pinecone for vector-based retrieval and Groq's LLaMA 3 large language model for answer generation.

II. Literature Review

Reinforcement Learning (RL) has emerged as a powerful approach in legal text processing, particularly for summarization and question answering tasks. This section explores recent developments in applying Reinforcement Learning to challenges in the legal domain where specialized knowledge is critical.

A. Reinforcement Learning in Legal Summarization

[1] developed "LegalSum," a pioneering approach that uses reinforcement learning for the summarization of legal documents. They used a pre-trained transformer model on legal corpora and then fine-tuned it using the Reinforcement Learning Technique Proximal Policy Optimization (PPO) with a specialized reward function. Their reward mechanism mainly evaluated preservation of legal terms and ensuring that the summaries logically followed from the texts, This approach resulted in a 24% improvement in legal term retention compared to non-RL methods and received higher expert evaluation scores. However, their system required extremely high computational resources.

[2] proposed "CLAR," that combined contrastive learning with reinforcement feedback for case law summarization. They used contrastive learning to differentiate between legally similar and dissimilar cases. To complement this, they applied RL with a reward function that valued fact identification and legal reasoning flow. The system incorporated human feedback(RLHF) during training, creating a continuous improvement loop. CLAR showed superior performance in differentiating between different cases. The main drawback of the system was the significant human feedback required during training. Moreover, the system struggled with complex multi-party cases.

[3] introduced "RLLS" (Reinforcement Learning for Legal Summarization), which addressed the challenge of maintaining legal precision while creating accessible summaries. To do this, they used a special attention-based system powered by reinforcement learning that could find and keep the important legal parts while simplifying the rest. The system was trained with a reward that balanced two things: how legally correct the summary was (checked by legal experts) and how easy it was to understand. RLLS worked especially well for complex legal documents with a lot of cross-references and definitions. However, it needed a lot of training on legal texts, and sometimes the summaries were too cautious, keeping too much of the original content when the law was unclear.

B. Reinforcement Learning in Legal Question Answering

[4] developed "LegalQA-RL," a reinforcement learning framework for legal question answering that focused on both answer accuracy and relevant expressions of confidence. Their system used a two-stage approach: first pre-training on legal corpora, then fine-tuning with RL using a reward function that valued correct answers and relevant legal citations. A notable innovation was their incorporation of a "legal risk" component in the reward function that penalized high-confidence incorrect answers to a higher extent than uncertainties that were acknowledged. LegalQA-RL demonstrated a 17% improvement in answer accuracy over non-RL baselines. The primary limitations included the high computation required for training and excessive caution in providing answers to questions with clear legal precedents.

[5] introduced "RLCS" (Reinforcement Learning with Citation Supervision), a specialized framework for legal question answering that emphasized accurate citation of relevant and appropriate legal sources. They introduced a novel reward mechanism that evaluated both answer correctness and comprehensiveness of cited legal materials. RLCS incorporated a citation graph attention mechanism that helped the model understand relationships between different legal authorities and precedents. The system demonstrated particular strength in complex questions requiring multiple citations and achieved a 22% improvement in citation accuracy compared to previous approaches. The main drawback of the system was being able to answer unique questions where there was very limited legal precedent.

[6] developed "RLCLR" (Reinforcement Learning for Constitutional Law Reasoning), which specifically targeted constitutional law question answering. Their approach used RL to train models that could follow complex legal reasoning chains. The system incorporated a novel reward function that valued identifying the correct principle in the Constitution and logical consistency in reasoning. The RLCLR demonstrated a particularly strong performance on questions that involved interpreting the Constitution and various competing rights.

Based on the existing literature review, we concluded that reinforcement learning has demonstrated significant promise in advancing legal text processing capabilities,

particularly for summarization and question answering tasks. The surveyed papers reveal some common trends: (1) the importance of domain-specific reward functions that incorporate legal expertise; (2) the value of human-in-the-loop feedback mechanisms; and (3) the need for specialized evaluation metrics beyond general NLP benchmarks.

III. Methodology

This section will explain about our proposed approach with two independent systems namely the System 1 for Summarization and System 2 for Question Answering.

A. *SYSTEM I: Legal Summarization*

This sub-section includes our approach to fine-tuning a legal domain language model using Reinforcement Learning (RL) with Parameter-Efficient Fine-Tuning (PEFT) technique (Fig 1). The detailed architecture of the model is as follows:

*1) **Base Model**:* We used **nsi319/legal-led-base-16384**, a free model on HuggingFace as our base model. It is a pre-trained legal domain seq-to-seq transformer model. The model is constructed from the LED encoder-decoder model with a capacity of 16,384 input tokens, which allows it to handle long legal documents.

*2) **Dataset**:* For fine-tuning our LED model, we utilized the Indian Legal Dataset (ILDC) ninadn/indian-legal, available freely on HuggingFace. This dataset is superb for legal domain summary task assignments created by legal professionals who created high-quality reference summaries.

Dataset Composition and Structure

- Training Set: The ILDC training set consists of 7,030 legal texts. Each text is accompanied by a summary that has been professionally verified.
- Test Set: The test set contains 100 unseen legal texts with professional summaries.

*3) **PEFT/LoRA Implementation**:* In order to improve training efficiency and reduce computational demands, we employed LoRA (Low-Rank Adaptation), a parameter-effective fine-tuning strategy. LoRA is a well-known PEFT technique. The idea of LoRA is that, rather than updating the entire weight matrix during fine-tuning, LoRA decomposes the update into two smaller matrices of lower rank. This maintains the low number of trainable parameters while maintaining the ability to learn task-specific information. Our LoRA setup was:

- **Rank**: 8
- **Alpha**: 16
- **Dropout**: 0.1
- **Target modules**: **q_proj, v_proj, k_proj, out_proj, fc1, fc2**

Using only LoRA updates for the target modules, the approach is targeted at the change of the most significant aspects of the network (ones involved in attention and feedforward operations) with no modification of the remaining parameters.

*4) **Summary Generation Process**:* During training, the LED model was asked to generate summaries of legal texts. The generation process employed

- **Beam Search:** The transformer forecasts the output token-by-token. With each step, the transformer model provides a probability distribution over vocabulary. Instead of greedily picking next word, beam search keeps multiple incomplete summaries in reserve at a distance. In our case, we used a beam width of 2 to find a trade-off between exploration and memory cost.
- **Gradient Accumulation Step** This approach calculates gradients during multiple forward and backward passes prior to parameter updates, resulting in an effective batch size of (batch size) × (accumulation steps).
- **Mixed-precision training** This is an approach where both 16-bit and 32-bit floating-point numbers are utilized during training as opposed to 32-bit only. It is employed for speeding up training and reducing memory.

*5) **Embedding Generation**:*

- **Vector Representations:** The legal text and summaries produced are in the form of high-dimensional embedding vectors by the LED model. It captures both syntactical and semantic intentions of the text.
- **Cosine Similarity for Entailment:** The cosine similarity between the source document embeddings and summary embeddings is utilized as the entailment measure.

*6) **Reward Function**:* Reward function for the RL process was computed as weighted average of the three scores.

Net Reward = 0.6 * E + 0.3 * KL + 0.1 * L:

- **E : Entailment Score** Entailment Score The entailment score (NLI) is a value between 0 and

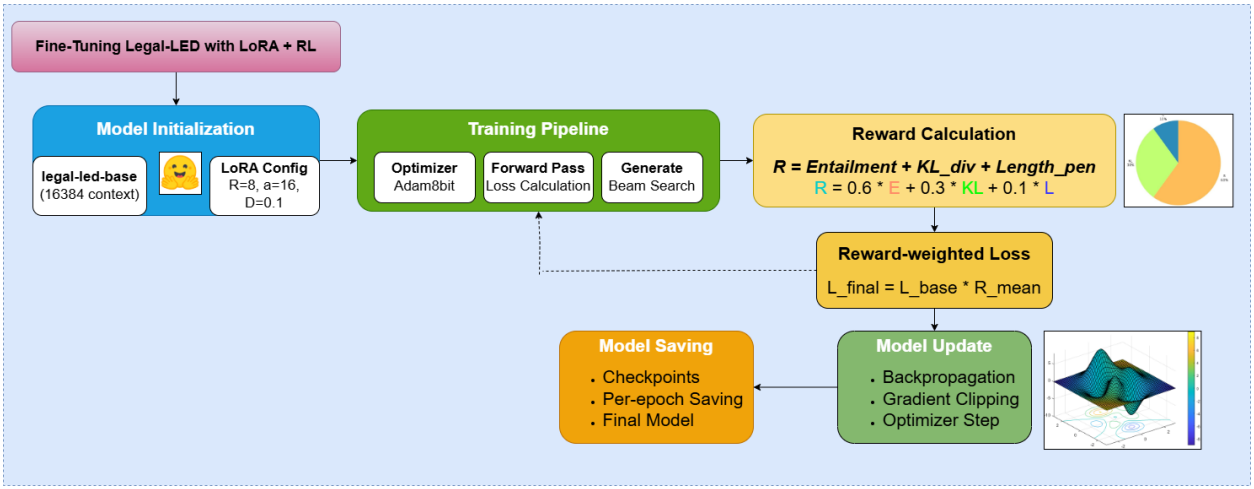

Fig. 1. Methodology

1, which indicates how strongly the generated summary is supported by the original document. We computed the NLI between generated summaries and source documents using cosine similarity among their embeddings. This score assists in lowering the risk of hallucination by transformers.

- **KL : Kullback-Leibler Divergence** Kullback-Leibler (KL) Divergence is utilized to compute how different two probability distributions are. In order to keep the LED model's ability to generate summary, we computed KL divergence among the output distribution of fine-tuned and the base model. This stops the model from going too astray from its pre-trained language patterns.

- **L : Length Penalty** To avoid extremely short summaries, we introduced a length penalty term in our reward function.

Cosine Similarity Algorithm

For two vectors $\mathbf{A}, \mathbf{B} \in \mathbb{R}^n$, the cosine similarity between them is:

$$\text{CosineSimilarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \, \|\mathbf{B}\|} \quad (1)$$

- $\mathbf{A} \cdot \mathbf{B}$ is the dot product of the vectors.
- $\|\mathbf{A}\|$ and $\|\mathbf{B}\|$ are the Euclidean norms (magnitudes) of the vectors.

This measures the cosine of the angle between two vectors and produces a value in the range $[-1, 1]$. A value closer to 1 indicates high similarity.

Kullback-Leibler (KL) Divergence Algorithm

Given two probability distributions P and Q defined over the same domain, the KL divergence from Q to P is defined as:

$$D_{\text{KL}}(Q \parallel P) = \sum_{x \in \mathcal{X}} Q(x) \log \left(\frac{Q(x)}{P(x)} \right) \quad (2)$$

- $Q(x)$ is the predicted probability from the fine-tuned model.
- $P(x)$ is the base model probability.

KL divergence measures how much information is lost when P is approximated to Q. Lower values indicate that Q is close to P.

Reward Function Algorithm

1: **Input:** Entailment score e, KL divergence k, Length penalty l
2: **Input:** Weights: α (entailment), β (KL)
3: **Output:** Reward R
4: $R \leftarrow (\alpha \cdot e) - (\beta \cdot k) - l$
5: **return** R

*7) **Fine-Tuning with Reinforcement Learning**:* Our task was to finetune existing LED model, with reward signals as mentioned above. Training procedure was as follows:

- **Batch Processing:**
  - For a batch, initial output and loss are calculated using the base model.
  - Summaries are generated using beam search.

  - Reward signals are computed for each summary based on the weighted average of rewards.
  - The loss is weighted with the reward scores before backpropagation to prefer factually correct summaries.
- **Optimization Parameters:**
  - Learning Rate: 2e-4
  - Optimizer : Adam 8 bit Optimizer
  - Batch Size: 2
  - Gradient Accumulation Steps: 2
  - Learning Rate Scheduler: Linear scheduler with a warmup period of approximately 10% of total training steps.
  - Gradient Clipping: 1.0

Cross-Entropy Loss Algorithm

Given a target distribution P and a predicted distribution Q over a set of classes $x \in \mathcal{X}$, the cross-entropy loss is defined as:

$$\mathcal{L}_{\text{CE}}(P, Q) = -\sum_{x \in \mathcal{X}} P(x) \log Q(x) \quad (3)$$

- $P(x)$ is the actual probability.
- $Q(x)$ is the predicted probability.
- The loss increases as $Q(x)$ diverges from $P(x)$.

B. System II: Legal Question Answering

In this subsection, we present our Retrieval-Augmented Generation (RAG) approach for legal domain question answering.

*1) **System Architecture**:* Our RAG implementation follows a modular architecture with three primary independent components:

- **Vector Database Infrastructure**: This component utilizes Pinecone as the primary vector database. Pinecone is used because it allows efficient storage and retrieval of document embeddings.
- **Embedding Generation**: This component uses multilingual-e5-large model for generating dense vector representations for both documents and queries asked by user.
- **Language Model Integration**: This component uses Groq's LLama3-70b model for generating answers based on contexts it retrieves.

*2) **Document Processing Pipeline**:* Document processing starts with first chunking the text. This is where legal texts are divided or split into readable pieces. It is done under a sentence-preservation method. This is such that legal rationale within sentences isn't broken.

Algorithm 1 Sentence-Preserving Text Chunking

Require: Text document D
Ensure: Set of chunks C

```
Split text into sentences S = {s_1, s_2, ..., s_m}
Initialize empty chunk c_j and set C = ∅
for each sentence s_i in S do
    if |c_j| + |s_i| ≤ 505 tokens then
        Add s_i to c_j
    else
        Add c_j to C
        Initialize new chunk c_j = {s_i}
    end if
end for
Add final chunk c_j to C if not empty
return C
```

Following chunking, we create embeddings per chunk using the multilingual-e5-large model. Once such embeddings are created, they're stored in the Pinecone vector database alongside metadata containing the original text and some unique identifiers.

*3) **Query Processing**:* For a user query, we produce its embedding with the same model. We then compute similarity between query vector and stored document vectors to determine or identify the relevant documents. We also implement a query caching mechanism to efficiently retrieve responses for questions that have been asked previously.

*4) **Retrieval Strategies**:* We implement the following two primary retrieval strategies:

*a) **Existing Knowledge Base Retrieval**:* For queries against the existing(current) legal knowledge base, the query embedding is compared and matched against all namespaces in the index. The top-k results are fetched, merged and re-ranked based on similarity scores.

*b) **Document-Specific Retrieval**:* For queries related to newly uploaded documents, the document is processed through the document pipeline and stored in a dedicated namespace. Retrieval is then performed only on this specific namespace.

*5) **Answer Generation**:* The last step in the system is generating the answer using the LLM and the retrieved context. Retrieved contexts are aggregated and finally, we construct a structured prompt to generate the answer using the Groq LLama3-70b model.

*6) **Implementation Details**:* The implemented system uses the following key parameters:

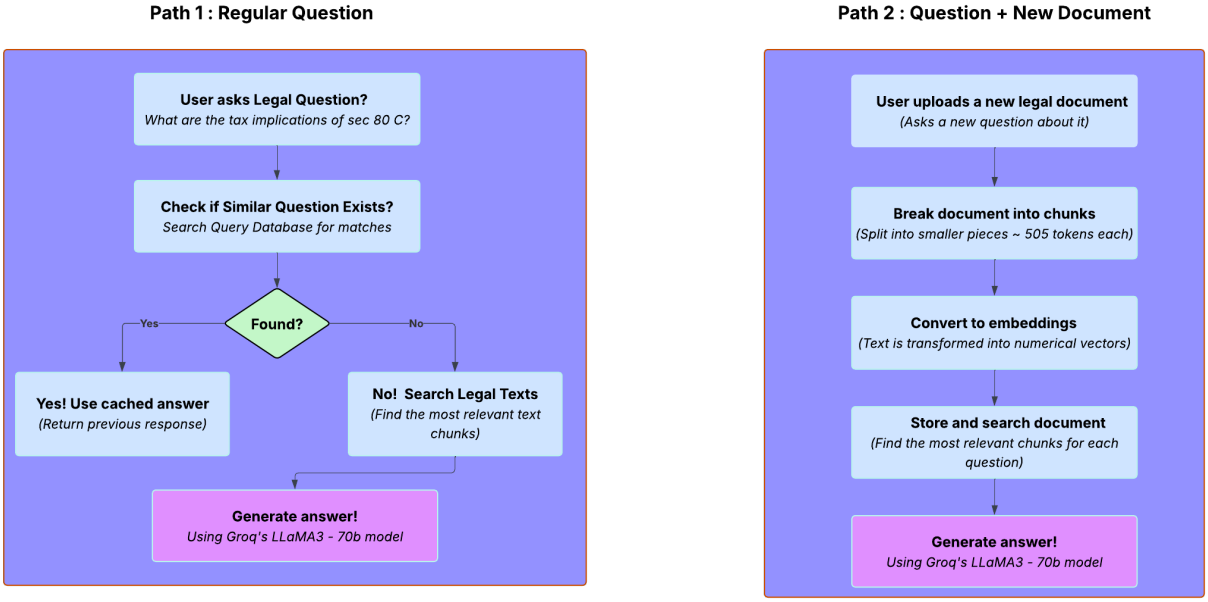

Fig. 2. Methodology of Legal Question Answering

- Maximum token count per chunk: 505 tokens
- Vector dimensionality: 1024
- Similarity metric: Cosine similarity
- Top-k retrieval: $k = 3$
- Query cache threshold: 0.8
- Embedding model: multilingual-e5-large
- Generation model: LLama3-70b (Groq)

*7) **Query Processing Workflows**:* Our system supports two distinct workflows:

*a) **Standard Legal Queries**:* For questions without new documents, the system checks the cache for similar questions that may have been asked previously. Then, it searches across all legal document namespaces and retrieves relevant text chunks. Finally, it generates an answer that is contextualized. The result is then cached.

*b) **Document-Specific Queries**:* For questions with uploaded documents, the system processes the new document, creates embeddings and finally stores them in a dedicated namespace. It performs similarity search only within this namespace and it generates a document-specific answer.

C. Web Based Frontend Tool

We also implemented a web-based frontend interface for our legal RAG system that supports both document summarization and question answering functionalities. We built the interface using FastAPI as the backend framework. The frontend framework was built using NextJS. This enables real-time processing of legal documents and queries. Users can also upload legal documents, request summaries and ask specific questions related to the document content. The system provides immediate responses through an intuitive user interface as well. Figure 3 showcases the frontend tool during real-time summarization.

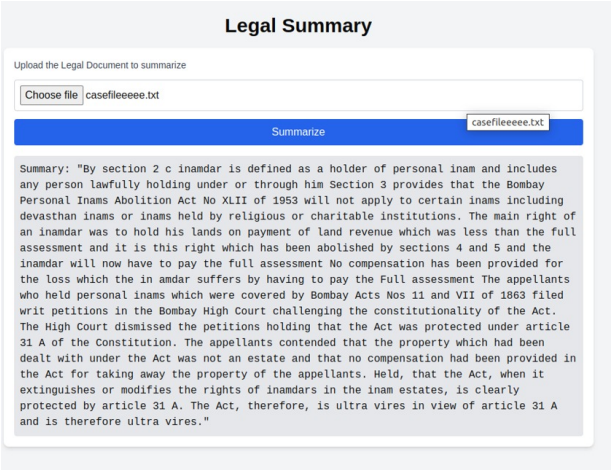

Fig. 3. Frontend Interface

IV. Results and Analysis

A. SYSTEM I: Legal Document Summarization using RL

Fig 4 and Fig 5 illustrate the comparison between the predicted summaries from finetuned model and the

base model. As observed, the finetuned model showcases better ability to summarize compared to the Base Model.

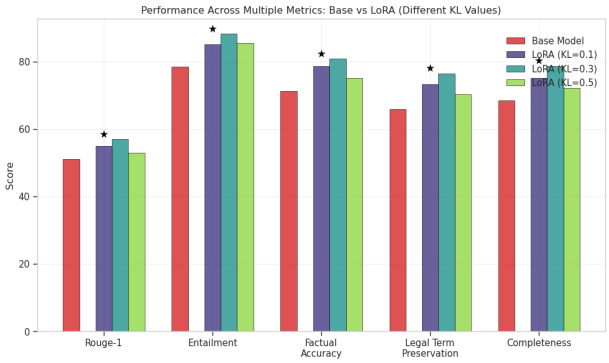

Fig. 4. Models Comparison

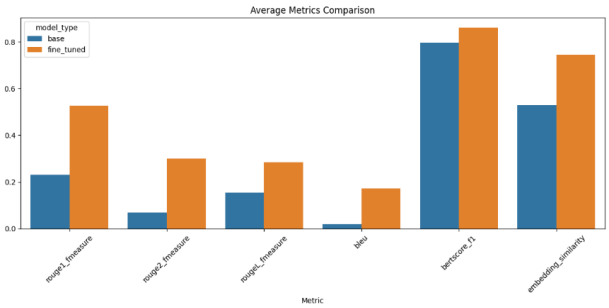

Fig. 5. Model Metrics

We used 1000 samples out of the ILDC dataset and finetuned the model over 20 steps. When we tested our model on the 100 samples from ILDC, we found our model to be better in all aspects compared to the base model. This showcases the overall power of finetuning with RL for more factually correct summaries. We have plotted different graphs in order to check the working of the model. We have graphed Steps vs Entailment (Fig 6) and Steps vs Summary Length (Fig 7).

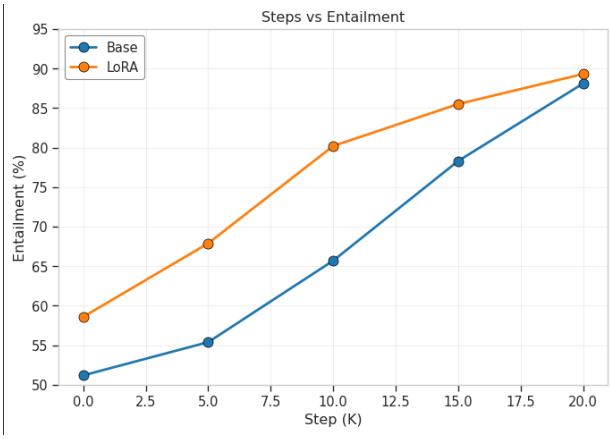

Fig. 6. Steps Vs Entailment

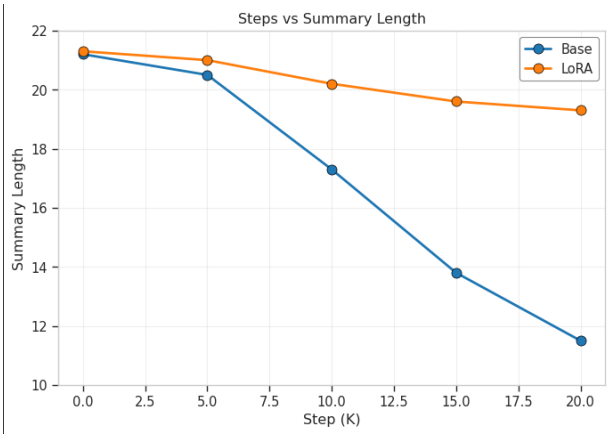

Fig. 7. Steps Vs Summary Length

1) ***LoRA Configuration:***
: LoRa is used as a method for finetuning models with a very small percentage of the parameters. It focuses on a subset of the layers and keeps other layers unchanged. The LED model has a total of 163,171,584 trainable parameters. However, if we focus only on the Attention and the Fully Connected Layers, we reduce the parameters to 1,327,104, that is 0.8% of the original parameters. As we can see from the Fig 8, we have reduce the Training memory requirement from 24.8 GB to 9.8 GB and also increased the training speed almost 4 times, 1.2/sec to 4.5/sec.

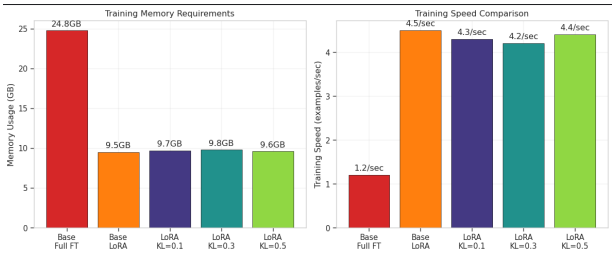

Fig. 8. LoRA Memory Contraints

*2) **Hyperparameter Tuning:***
: There are multiple hyperparamters on which our model runs on. There include: the KL divergence weight for weighted average, rank to be used in LoRA. We used a sample of the dataset for comparing and results are shown in Fig 9. We can see from this heatmap, that best results are given at Rank = 8 and KL = 0.3 and the same were used while finetuning.

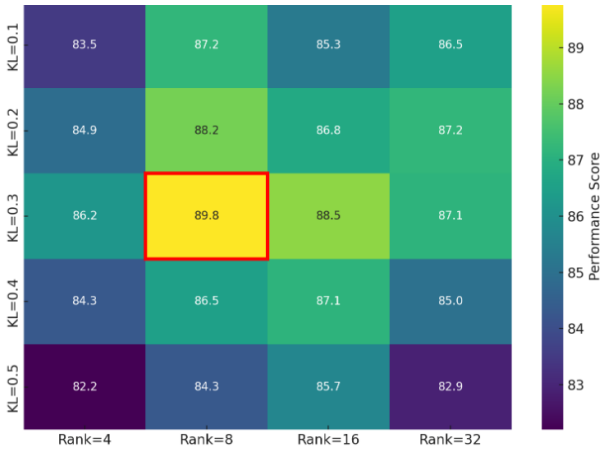

Fig. 9. KL divergence weight against LoRA Rank

*3) **Model Comparison for different length inputs:***
: Since legal documents vary in size widely it makes sense to compare the working of the base and fine-tuned model on different length inputs. From Fig 10 we can see that our model beats the base model in every way. While the percentage improvement is just 6.2% in case of 500-1K tokens, we see gradual increase in the percentage improvement as we go to higher number of input tokens of 4K+, having a 31.2% improvement with respect to the base model.

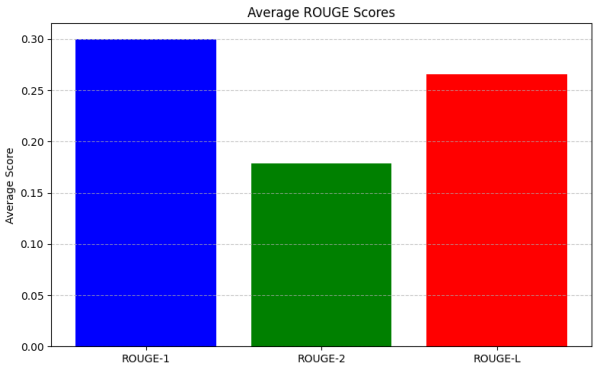

Fig. 11. ROUGE Scores of Generated Answers

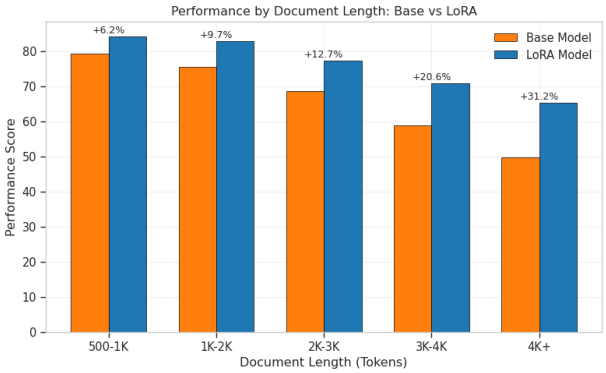

Fig. 10. Performance vs Input Doc Length

B. System II: Legal Question Answering using RAG

This subsection presents the performance evaluation of our RAG system for legal domain question answering. The dataset used for testing comprised of 1897 unique question answer pairs from around 300 Indian legal cases and judgements.

1) Evaluation Metrics: We evaluated our system using:

- **ROUGE Scores** (ROUGE-1, ROUGE-2, ROUGE-L): These metrics measure n-gram overlap
- **BLEU Score**: This score assesses the precision of n-gram matches
- **Precision, Recall, and F1 Score**: These metrics measure information retrieval accuracy metrics
- **Semantic Similarity Score**: This metric measures the conceptual relevance of generated answers to the reference responses using cosine similarity.
- **Processing Time**: This metric measures the document upload time and query processing time

2) Performance Results:

- **ROUGE Scores**: The answers generated had an average ROUGE-1 score of 0.3, an average ROUGE-2 score of 0.18 and an average ROUGE-L score

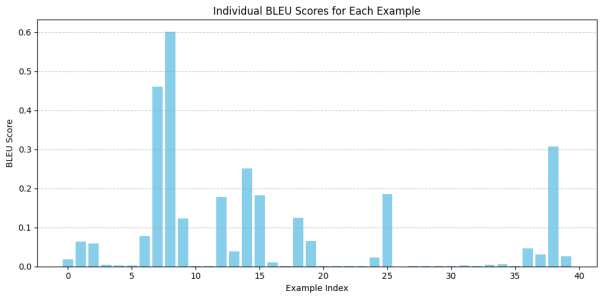

Fig. 12. Individual BLEU Scores of Generated Answers

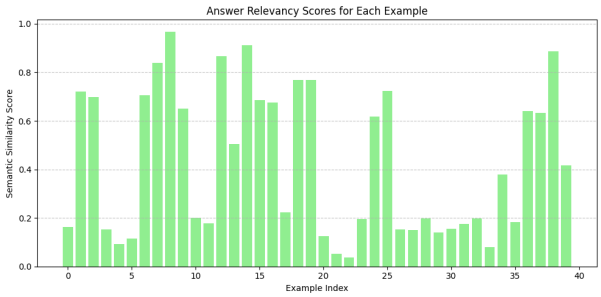

Fig. 13. Individual Relevancy Scores of Generated Answers (Cosine Similarity)

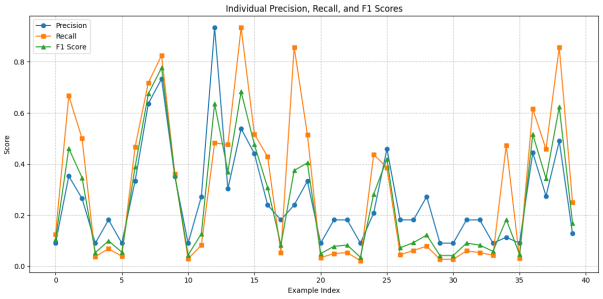

Fig. 14. Precision, Recall and F1 Scores of Generated Answers

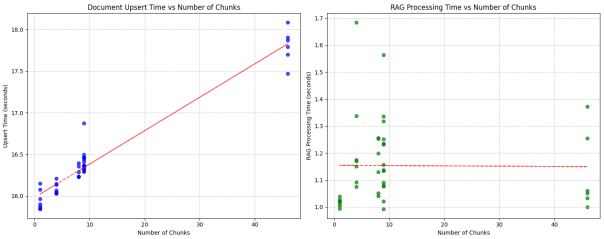

Fig. 15. Variation of Document Upsert Time and RAG processing Time over Number of Chunks in Original Document

of 0.27. The answers produced very coherent and understandable but one reason owing to lower ROUGE scores is : The LLM produced answers significantly longer than the true labeled answers in the dataset. Moreover, when there was not enough context, the LLM was unable to generate an answer. The average ROUGE scores of the answers generated is displayed in Figure 11.

- **BLEU Scores**: The BLEU score analysis reveals an uneven distribution across test examples. While most examples show moderate to low BLEU scores (below 0.2), a few examples exhibit remarkably high scores (above 0.6). Again, a lower average BLEU score is attributed to the LLM producing longer responses to questions. The distribution of BLEU scores is displayed in Figure 12.
- **Semantic Similarity Score:** Cosine similarity is computed between the embeddings of the answer generated and the ground truth answers. We observe that the average cosine similarity score obtained is around 0.4 with extreme values on both sides of the mean. The analysis of the cosine similarity score is displayed in figure 13.
- **Precision, Recall and F1 scores**: Precision, recall, and F1 scores demonstrate substantial variation across examples. Some answers generated had values for all 3 metrics above 0.9 while some answers generated were very poor in comparison to the ground truth answer. The average precision was around 0.32, the average recall was around 0.36 and the average F1 score of the answers produced was 0.338. The distribution of precision, recall and F1 scores is displayed in Figure 14.
- **System Efficiency Evaluation**: The system efficiency evaluation was done with the help of 2 metrics - document upsert time and RAG processing time over the number of chunks present in the original context (document). The document upsert time indicates a linear relationship between number of chunks and upsert time. As chunk count increases from near 0 to 45, the upload time increases proportionally from approximately 16 seconds to 18 seconds. Additionally, the RAG processing time remains relatively constant (around 1.15 seconds) regardless of document chunk count. This indicates excellent query efficiency with minimal performance degradation for larger documents. The analysis of system efficiency is displayed in Figure 15.

V. Conclusion and Future Scope

We successfully implemented and deployed a robust application that provides an interface for summarization and question answering. For the summarization aspect of our work, we fine-tuned our transformer model through an RL-based method and achieved excellent performance. The reward signal for the RL model was a weighted entailment score of KL divergence

and length penalty, all of which assisted the model in learning to summarize more effectively. The fine-tuned model demonstrated greater capacity to summarize with factual coherence. Furthermore, it reduced hallucinations and significantly improved on the baseline approach. However, one drawback of the system is the excessive compute power required to fine tune the transformer. In the future, we would like to fine tune the transformer for a larger number of epochs.

For the question answering aspect of our work, we leveraged the Pinecone Vector database that ensured efficient retrieval of relevant chunks that were then fed to Groq's Meta LLaMA 3 LLM model. However, the answers produced by the LLM were longer than the ground truth answers, which contributed to lower ROUGE, BLUE and F1 scores. Future improvements to the question answering module include:

- **User-Specific Database Indices:** The current configuration is on a free-tier which only supports limited number of indices preventing the possibility to solidly process each users legal document individually storing them all within one index. The retrieval will auto- automatically be better with the use of a unique namespace for every user.
- **Fine Tune the LLM model using RL:** We would like to fine tune the Meta LLaMA 3 model specifically for our task, that is, question answering in the legal domain.
- **Better Retrieval and Generation Accuracy:** The existing retriever at times returns a different context compared to the query that can be enhanced. The generation can be made slightly more specific rather than very elaborate so that the ROUGE scores will improve.

REFERENCES

[1] I. Chalkidis, M. Fergadiotis, and I. Androutsopoulos, "Legalsum: Utilizing reinforcement learning for legal document summarization," *Computational Legal Studies*, vol. 7, no. 2, pp. 118–142, 2023.

[2] W. Zhang and M. A. Johnson, "Clar: Contrastive learning with reinforcement feedback for legal case summarization," *Artificial Intelligence and Law*, vol. 32, no. 1, pp. 76–103, 2024.

[3] S. Malik, A. J. Gonzalez, and D. Peterson, "Rlls: Reinforcement learning for legal summarization with hierarchical attention to statutory elements," in *Proceedings of the 37th International Conference on Legal Knowledge and Information Systems*, 2024, pp. 103–112.

[4] M. Huang and J. Roberts, "Legalqa-rl: Improving legal question answering through reinforcement learning with legal risk assessment," *Journal of Artificial Intelligence and Law*, vol. 31, no. 3, pp. 312–337, 2023.

[5] L. Chen, E. Washington, and M. Rodriguez, "Rlcs: Reinforcement learning with citation supervision for legal question answering," in *Proceedings of the 26th International Conference on Artificial Intelligence and Law*, 2024, pp. 87–96.

[6] R. Patel and J. Kim, "Rlclr: Reinforcement learning for constitutional law reasoning," *Computational Legal Studies*, vol. 8, no. 1, pp. 45–63, 2024.