NeurIPS-2025 - RankLLaMA Based Tree-RAG Using Lightweight Models for Answering Robotic-Assisted Surgery Queries - Supplementary materials

May 22, 2025

Repository and Documentation

Here we present all the README files included in our repository for reference. However, we encourage users to directly consult and use our official GitHub repository for the most up-to-date and organized access to the materials.

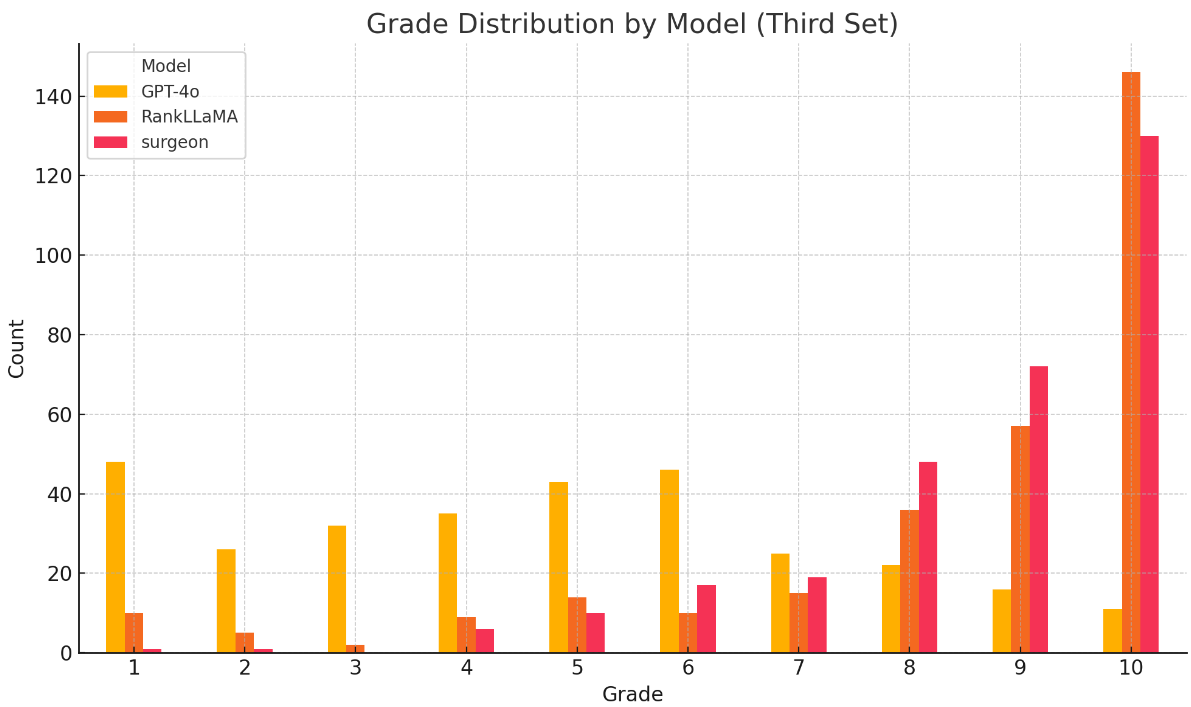
Access the full repository here (recommended): link to the repository

NeurIPS_2025 Repository

This repository provides all the code used for the research: *RankLLaMA-Based Tree-RAG Using Lightweight Models for Answering Robotic-Assisted Surgery Queries*. We plan to publicly share all documents upon paper acceptance.

Included Files and Descriptions

- **NeurIPS_2025.xlsx**: Contains the benchmark, a subset of our dataset, and the corresponding questions from the textbook (*RAS QA Sample*, *ras_texbook_sample*), as well as the four QA sets submitted to the surgeons (original version, blind version used by the surgeons, the mapping between the two, and the grades reported in Fig. 4 of the manuscript).



- **ras_qa_rag_eval.xlsx**: Contains the evaluation results for all models discussed in Table 1. A column named *retrieved content* (an array of all retrieved segments) must be removed, as the full dataset could be reconstructed from it. These tables were generated using the RAGAS framework—please refer to Subsection 4 for more details.

Table 1: Evaluation of models using Cosine Similarity and RankLlama + Tree-RAG.

	Model Info		Context Precision	Context Recall	Faithfulness	Answer Relevancy	Semantic Similarity	Time (mean)	Time (total)
	Model	Dimension/Size							
Cosine Similarity	Linq-Embed-Mistral [39]	4096	0.7651	**0.9092**	0.8598	0.7442	0.7713	1.9	596.5
	multilingual-e5-large-instruct [40]	1024	0.6857	0.8219	0.8631	0.7547	0.7714	1.8	**558.1**
	jina-embeddings-v3 [41]	1024	0.6619	0.7828	**0.9167**	0.6912	0.7656	5.5	1665.4
Mean	—	—	0.7042	0.8380	**0.8799**	0.7300	0.7694	3.1	940.0
Std. dev.	—	—	0.0540	0.0647	0.0319	0.0340	0.0033	2.1	628.5
RankLlama + Tree-RAG	Llama-3.2-1B-Instruct [42]	1.24B	0.8725	0.8518	0.8388	0.7730	**0.8072**	12.3	3814.4
	Llama-3.2-1B-Instruct_st15 [42]	1.24B	0.8674	0.8414	0.8107	0.9016	0.7890	8.9	2774.5
	Qwen2.5-1.5B-Instruct [43]	1.54B	**0.8918**	0.8579	0.7569	**0.9485**	0.7793	16.2	5015.3
	gemma-3-1b-it [44]	1B	0.8798	0.8580	0.8845	0.7352	0.7974	15.5	4802.1
	Llama-3.2-3B-Instruct [42]	3.21B	0.8760	0.8554	0.8851	0.7654	0.7983	13.3	4132.3
	Qwen2.5-3B-Instruct [43]	3.09B	0.8768	0.8555	0.8309	0.8369	0.7699	17.9	5540.9
	gemma-3-4b-it [44]	4.3B	0.8794	0.8383	0.8797	0.8090	0.7972	18.6	5755.1
	Mistral-7B-Instruct-v0.3 [45]	7.25B	0.8808	0.8573	0.8811	0.8938	0.7853	16.9	5242.1
	Llama-3.1-8B-Instruct [42]	8.03B	0.8835	0.8530	0.8741	0.8548	0.7938	16.6	5140.5
	Qwen2.5-7B-Instruct [43]	7.62B	0.8778	0.8518	0.8625	0.8780	0.7823	16.8	5214.4
Mean	—	—	**0.8786**	**0.8520**	0.8504	**0.8396**	**0.7900**	15.3	4743.1
Std. dev.	—	—	0.0065	0.0069	0.0418	0.0684	0.0110	2.9	911.6

- **ras_qa_sample**, **ras_texbook_sample**: Also included in *NeurIPS_2025.xlsx*, these are subsets of our dataset (specifically covering three procedures) made available to support the validity of our work.

- **tree_creation_book.py**, **RAG_NeurIPS.py**: The RankLLaMA Tree-RAG code described in Subsection 2.3. These scripts were used to generate the evaluation inputs for the RankLLaMA+TreeRAG results in Table 1 of the manuscript.

- **RAG_cos_NeurIPS.py**, **embedding.py**: The equivalent scripts for the "classic" RAG evaluation used in Table 1 (Cosine Similarity).

Table 2: Nvidia Answer Accuracy Across Models

Metric	Linq-Embed-Mistral	GPT-4o	Mistral-7B-Instruct-v0.3
Nvidia Answer Accuracy	0.6634	0.4942	0.8133

Measurement Details

We provide additional insights into the details of our measurements:

- **Table 1**: The "Var" row was created using the `=STDEV(column above)` function in Excel (Gamma3). For **Context Precision**, the reported variance reflects the variability of the metric itself, as it is model-agnostic (cf. Eq. 7, Subsection 4.1). However, for **Context Recall**, the same conclusion cannot be drawn (cf. Eq. 9, Subsection 4.1).

- **Table 2**: More details in the folder RAG Evaluation

Cloning the Repository

```
git clone https://github.com/neuripsapplication/1o3u9u324iu2
cd /1o3u9u324iu2
```

Fine-Tuned Models

We also provide two preliminary fine-tuned models (**LLaMA-3.3-70B**), trained on our benchmark (note: these models have not been evaluated): link to the models

- **qa_generator_llama3.3-70B**: A question generator that, given a chunk of text, generates a specialized question on the topic.
- **surg_qa_LLaMA-3.3-70B-Instruct**: A question-answering model that, given a RAS query, returns an appropriate answer.

The fine-tuning conditions are described in Subsection 4.3 *Preliminary Fine-Tuning*.

The code was tested on the following hardware configuration: **One AMD EPYC 7742 64-Core Processor, 8 Nvidia-A100 GPUs (40 GB)**.

RAG

Hardware Specifications

The code was tested on the following hardware configuration:

- One AMD EPYC 7742 64-Core Processor
- One Nvidia-A100 GPU (40 GB)

Installation Instructions

To set up the environment and install the required dependencies:

Create and activate a virtual environment:

```
python3 -m venv run_rag
source run_rag/bin/activate
pip install -r requirements.txt
```

How to Run

- **RankLLaMA+Tree-RAG**: Open the `RAG_NeurIPS.py` script and follow the instructions at the top of the file. Run the script to see a demonstration of our retrieval method on the shared subset of the dataset (`ras_qa_sample`, `ras_texbook_sample`).
- **Cosine Similarity (Classic RAG)**: Open the `embedding.py` script and follow the instructions provided at the top. Then, execute both `embedding.py` and `RAG_cos_NeurIPS.py` to reproduce the baseline results.

Evaluation

Hardware Specifications

The evaluation code was tested on the following hardware:

- One AMD EPYC 7742 64-Core Processor
- Two Nvidia A100 GPUs (40 GB each)

Installation

To set up the virtual environment and install the dependencies:

```
python3 -m venv eval_rag
source eval_rag/bin/activate
pip install -r requirements.txt
```

Before running the evaluation scripts, ensure that Ollama is installed and actively running in the terminal.

`evaluation_RAG_ragas.py`

This script uses a Judge LLM (Gemma3-27B via Ollama) and an embedding model (jina-embeddings-v3) to evaluate standard RAG metrics as described in the RAGAS documentation.

CSV files evaluated:

- `rag_cos_jina_sample.csv`: Classical RAG with jina-embeddings and answer generation using Llama-3.2-1B-Instruct.
- `rag_mistral7B_sample.csv`: RankLLaMA+Tree-RAG retrieval with answer generation using Mistral-7B-Instruct-v0.3.

These files include QA pairs for three surgical procedures from the textbook. Evaluation outputs such as `rag_cos_jina_sample_ragas_eval.xlsx` and `rag_mistral7B_sample_ragas_eval.xlsx` are provided. Reported average metrics ($\pm 1\sigma \approx 0.05$) are:

Table 3: RAGAS Evaluation Results

Model	**Context Precision**	**Context Recall**	**Faithfulness**	**Answer Relevancy**	**Semantic Similarity**
Mistral-7B-Instruct-v0.3	0.9796	0.9566	0.8517	0.9340	0.7363
jina-embeddings-v3	0.4302	0.6051	0.6539	0.7565	0.7223

`evaluation_RAG_nvidia.py`

This script evaluates the same models on Nvidia-specific QA using the file `ras_qa_nvidia_eval.xlsx`, which contains three sheets corresponding to:

- GPT-4o
- Linq-Embed-Mistral + Llama-3.2-1B
- RankLLaMA + Tree-RAG + Mistral-7B

The user can select a model within the script to evaluate and generate output files (e.g., `rag_qa_nvidia_eval_GPT-4o.x`

Reproducibility: Each model was evaluated across five independent trials. The mean and standard deviation are reported:

Table 4: Nvidia Evaluation Trials

Model	**Trial 1**	**Trial 2**	**Trial 3**	**Trial 4**	**Trial 5**	**Mean**	**Std. Dev.**
Mistral-7B-Instruct-v0.3	0.8133	0.8127	0.8144	0.8144	0.8127	0.8135	0.0009
GPT-4	0.4942	0.4983	0.4975	0.4983	0.4975	0.4972	0.0017
Linq-Embed-Mistral	0.6634	0.6337	0.6328	0.6328	0.6320	0.6389	0.0137