First, ensure you have a workstation equipped with at least a 12 GB NVIDIA GPU and that Python 3.8 is installed under a Conda-managed environment. Create and activate a fresh environment with

**conda create -n llm\_summarize python=3.8 -y**

**conda activate llm\_summarize**

**pip install pdfplumber python-docx**

Then download and configure the Ollama CLI from <https://ollama.com> so that the ollama command is available in your shell.

Next, pull the DeepSeek-14B model into your local Ollama repository by running

**ollama pull deepseek-r1:14b**

Once the download completes, verify the installation with

**ollama list**

**ollama run deepseek-r1:14b "Hello, please generate a brief test sentence."**

If you require finer control over inference parameters or wish to use the Hugging Face transformers interface, export the model via

**ollama export deepseek-r1:14b ./exported\_deepseek**

You will then find a directory exported\_deepseek/ containing config.json, tokenizer.json, pytorch\_model.bin, and other necessary files for direct loading.

We provide a script named summarization\_cli.py that orchestrates the entire workflow: it detects whether your input is a PDF, DOCX, or TXT file and uses pdfplumber or python-docx to extract all text into a single string. It then splits that string into manageable chunks based on paragraph structure and character-length thresholds, invokes ollama run deepseek-r1:14b on each chunk to produce intermediate summaries, and finally concatenates (and, if needed, re-summarizes) these snippets into one coherent Chinese summary.

To run the script, place it alongside your documents and execute:

**python summarization\_cli.py path/to/your\_document.pdf --model deepseek-r1:14b**

The script will print the first 500 characters of the original text for quick verification, report progress on each chunk’s summarization, and then display the complete “Final Summary.” No manual splitting or intervention is required—just one command to generate a full document summary.

After you obtain summaries, evaluate their quality on a small test set of 10–20 documents using a tool like py-rouge, aiming for a ROUGE-L score of at least 30. You may also conduct a blind human evaluation with 1–2 peers who rate 5–10 summaries on fluency, accuracy, and information coverage, targeting an average score of 3.5 out of 5. If you find the summaries need refinement or performance improvements, adjust the prompt wording in the script, add inference options such as --temperature or --top-p to the ollama run invocation, or switch to 4-bit quantization to reduce memory usage and speed up inference.