MMPROVENCE: TURNING A MULTIMODAL RERANKER INTO A ZERO-COST CONTEXT PRUNER

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal RAG pipelines pass the retrieved pages to a vision-language model (VLM) as images together with their text, and the page images make up most of the model's input, yet existing context pruners operate on text only. We introduce MMProvence, a zero-cost context pruner for multimodal RAG: a VLM reranker that, in a single call per query–page pair, generates a relevance score, a bounding box on the page image, and the page sentences that answer the query. The score reranks the candidates, and the box and the sentences prune the image and the text of the pages that survive reranking, so pruning requires no additional model call. We further show that what the box is trained to cover decides the cost of cropping: on top of text pruning, crops to boxes that include the context that makes the evidence readable, such as table headers or chart legends, lower the LLM-as-a-judge score by 2.7 points, against 13.3 points for boxes around the smallest answer-bearing region. On the eight domains of ViDoRe V3, MMProvence-2B reduces the generator's input by 75%, from 21.6k to 5.3k tokens, at an LLM-as-a-judge score of 0.803 against 0.827 with the full context, while ranking on par with the reranker it replaces (nDCG@10 of 0.601 vs. 0.604). At 4B and 9B parameters, MMProvence ranks better than this reranker (0.627 and 0.636), and MMProvence-9B keeps a score of 0.822 while removing 76% of the input. We will release our code, checkpoints and training data, together with VERGE, our framework for end-to-end multimodal RAG evaluation.

1 INTRODUCTION

Large Language Models (LLMs) are trained on data collected up to a cutoff date, yet they are routinely asked about knowledge beyond it, or about private collections they have never seen. Retrieval-Augmented Generation (RAG) addresses both limitations: it retrieves documents relevant to the query and feeds them to the LLM together with the query, grounding the answer in up-to-date, domain-specific evidence and reducing hallucinations (Lewis et al., 2020). In enterprise settings, much of this evidence lives in visually rich documents such as reports, slide decks, manuals and forms whose content is carried by tables, charts and layout as much as by running text. Visual document retrieval therefore indexes the page *images* directly (Faysse et al., 2025; Yu et al., 2025), and multimodal RAG pipelines pass the retrieved pages to a Vision-Language Model (VLM) as images, often together with their extracted text (Loison et al., 2026).

However, the benefits of RAG come with significant computational costs. Retrieved documents dominate the input length, and in multimodal RAG they are dominated in turn by page images: in the pipeline we study (Section 5), the VLM reads on average 21.6k input tokens per query for five retrieved pages, and **82.4%** of them are image tokens (Figure 1b). Since self-attention scales as $\mathcal{O}(N^2)$ in the input length N (Vaswani et al., 2017), both in the language model and in the vision encoder, this cost grows faster than the context itself. Conversely, every token removed before generation saves more than its proportional share of compute. Context pruning, the selective removal of retrieved content that is irrelevant to the query, has shown particular promise for reducing this cost in text RAG (Jiang et al., 2023; Xu et al., 2024; Hwang et al., 2024; Chirkova et al., 2025). Yet the context pruners developed for RAG operate on text alone: even one that deleted every retrieved sentence would leave more than four fifths of the VLM's input untouched.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

A state-of-the-art approach in this space is Provence (Chirkova et al., 2025), which makes pruning *zero-cost* by integrating it into the reranking stage of the RAG pipeline: a lightweight head on the reranker labels each sentence of a passage as relevant or irrelevant, in the same forward pass that scores the passage. XProvence (Mohamed et al., 2026) extended this design to more than 100 languages. Both, however, prune text only, and their mechanism does not transfer to images, which have no sentences to classify. Pruning a page image means *cropping* it to the region the answer is read from. This is a localization problem with a failure mode of its own: a crop can keep a value while cutting off the column header or the legend that says what the value measures.

In this work, we introduce MMProvence, a zero-cost multimodal context pruner for RAG over visually rich documents. MMProvence is a VLM reranker that, for each query-page pair, generates a single JSON object holding a relevance score, a bounding box on the page image, and the page sentences that answer the query, copied verbatim (Figure 1a). The score ranks the candidates; for the pages that survive reranking, the box crops the image and the sentences replace the text. Since the pruning decisions are produced by the reranking call itself, the pruning stage issues no model call at all. Our contributions can be summarized as follows:

- **Zero-cost multimodal context pruning:** We introduce MMProvence, which prunes both the image and the text of retrieved pages as a by-product of reranking. We train it from the general-purpose Qwen3.5 VLM (Qwen Team, 2026) at three scales (2B, 4B and 9B), by distilling the relevance scores of Qwen3-VL-Reranker-8B (Li et al., 2026) into it while it learns to point at its evidence, on a mixture of text, natural-image and document-page data annotated with boxes and sentences. We also explore adding pruning to the Qwen3-VL-Reranker-2B cross-encoder through LoRA fine-tuning (Appendix E).
- **Contextual evidence boxes:** We show that what the box is trained to cover decides how much cropping costs. Added on top of text pruning, crops to boxes trained on the smallest answer-bearing region lower the LLM-as-a-judge score by 13.3 points, whereas crops to boxes trained to also cover the context that makes the evidence readable, such as the headers of a table cell or the legend of a chart, lower it by 2.7 points. Applied alone, contextual crops remove about three quarters of the page area and lower the score by at most 0.7 points.
- **Extensive evaluation on 8 domains:** On the eight public domains of ViDoRe V3 (4,000 queries), MMProvence-2B cuts the VLM's median input from $21.6k$ to $5.3k$ tokens (-75%) at an LLM-as-a-judge score of 0.803, against 0.827 with the full context. MMProvence-2B ranks on par with the Qwen3-VL-Reranker-2B cross-encoder it replaces (nDCG@10 of 0.601 vs. 0.604), and MMProvence-4B and -9B rank better than it on all eight domains (0.627 and 0.636), so pruning does not come at the expense of reranking.
- **A high-performance framework for end-to-end multimodal RAG evaluation:** We implement VERGE (Visual End-to-end RAG Evaluation), which runs a multimodal RAG pipeline from retrieval over page images to the judged answer. Following BERGEN (Rau et al., 2024), every component is a configuration file behind a shared interface, and VERGE serves every model with vLLM (Kwon et al., 2023), caches every stage, runs the LLM judge in parallel across GPUs, and records per-stage latency and token counts (Appendix B). We will open-source VERGE, our code, checkpoints and training data.

2 Background and Related Work

Reranking stage of the RAG pipeline. Production-ready RAG systems typically consist of three main components: a retriever, which coarsely identifies relevant documents in a datastore; a reranker, which refines the selection; and a generator, which produces the final answer (Rau et al., 2024). Rerankers score each candidate by encoding it jointly with the query, which is more expensive than bi-encoder retrieval but more accurate, with architectures ranging from BERT-style cross-encoders (Nogueira & Cho, 2019) to point-wise, pair-wise and list-wise LLM rerankers (Zhuang et al., 2024; Qin et al., 2024; Sun et al., 2023). In multimodal RAG, the reranker is a VLM that reads the query together with a page image and its text: Qwen3-VL-Reranker (Li et al., 2026), for example, is prompted to judge whether the document meets the query and scores it as $\sigma(l_{\text{yes}} - l_{\text{no}})$, from the logits of the `yes` and `no` tokens. Reranking has high leverage on answer quality: in our

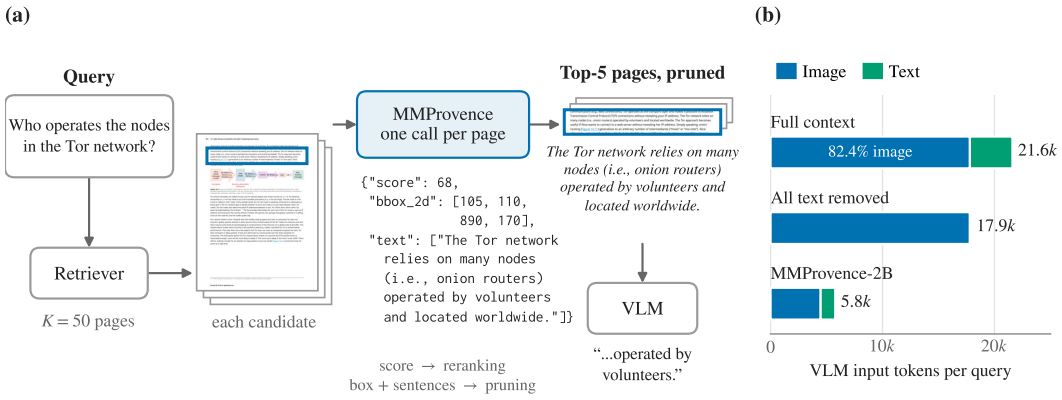

Figure 1: MMProvence prunes multimodal RAG contexts at zero cost. (a) For every query–page pair, one generation returns a relevance score, a bounding box and the verbatim relevant sentences, shown here for a ViDoRe V3 query and the output of MMProvence-2B. The score reranks the retrieved pages, and each page kept for generation is cropped to its box and cut down to its sentences, with no additional model call. (b) VLM input tokens per query, averaged over the 4,000 queries. With the full context, page images account for 82.4% of the input, so even deleting all of the text leaves 17.9k tokens, whereas MMProvence-2B leaves 5.8k (median: 5.3k).

pipeline, handing the retriever's top-5 pages straight to the generator lowers the LLM-as-a-judge score from 0.827 to 0.761 (Section 6).

Provence and zero-cost context pruning. Provence (Chirkova et al., 2025) enhances a cross-encoder reranker with context pruning by applying an additional linear head to the representations of the passage tokens. The head outputs a per-token relevance probability, which is binarized with a threshold, and a sentence is kept if more than half of its tokens are relevant. It is trained with a per-token cross-entropy objective, whose targets come from an LLM prompted to answer the query from the passage while citing the sentences it used, plus a regression objective that preserves the reranker's original scores. Because the relevance score and the pruning decisions come from the same forward pass, pruning adds no cost to a pipeline that already reranks. XProvence (Mohamed et al., 2026) extended this recipe to more than 100 languages. Other text pruners and compressors follow various strategies: (i) select sentences independently of each other (Xu et al., 2024; Hwang et al., 2024), (ii) drop tokens by their perplexity under a small language model (Jiang et al., 2023), or (iii) summarize the context (Xu et al., 2024); all of them require a model call of their own, and all of them operate on text.

Multimodal RAG over visually rich documents. Rather than parsing documents into text, visual document retrievers embed page screenshots directly, either as a single vector (Ma et al., 2024; Li et al., 2026) or as multiple patch-level vectors scored by late interaction (Faysse et al., 2025). Retrieved pages are then passed to a VLM as images (Yu et al., 2025; Cho et al., 2024), which preserves the tables, charts and layout that text extraction loses. In practice, the image is usually accompanied by the page's extracted text, obtained by OCR or document parsing: the text is cheap to include and gives the VLM exact strings and numbers that are hard to read reliably off a rendered page. The ViDoRe benchmarks (Faysse et al., 2025; Loison et al., 2026) evaluate this setting, and ViDoRe V3, which ships such a transcription for every page, adds human-verified answers, so that end-to-end answer quality can be measured. Keeping the page as an image comes at a price: in our pipeline, a single page costs the generator about 3.6k visual tokens at its native resolution, against 0.8k tokens for the page's extracted text.

Visual token reduction. A separate line of work reduces the number of visual tokens inside the VLM itself, e.g., by discarding tokens that receive little attention after the first layers (Chen et al., 2024) or tokens that are irrelevant to the text prompt (Zhang et al., 2024). These methods are tied to the internals of a specific generator and still run the vision encoder on the full image. MMProvence instead crops the page image before the generator receives it, so the reduction is independent of

the generator's architecture and weights, and also removes vision-encoder compute. **The two approaches are complementary and can be used together**: MMProvence decides which region of each page reaches the generator, and token reduction can thin out the visual tokens of that crop without changing either method; we leave measuring the combination to future work.

3 MMPROVENCE

For a query q, the retriever returns K candidate pages, each consisting of a page image and its extracted text. The reranker assigns every candidate a relevance score, and the k highest-scored pages are passed to the generator ($K = 50$ and $k = 5$ in our experiments, Section 5). MMProvence takes the place of the reranker. It is a VLM that, for each query–page pair, generates a single JSON object with three fields: a relevance score `score`, an integer from 0 to 100; a bounding box `bbox_2d` on the page image, as four integers on a 1000×1000 grid; and a list `text` of page sentences copied verbatim. The score reranks the K candidates, and for the k pages that survive, the box and the sentences prune the image and the text (Figure 1a).

Input and output format. The full prompt is given in Appendix H (Figure 5), and Figure 1a shows an output. The system turn opens with the instruction of Qwen3-VL-Reranker (Section 2), extended to ask for the evidence, and then gives the schema and a rule for each field. The user turn holds the query and the document, i.e., the page image followed by the page text. The assistant turn is prefilled with an empty reasoning block, which disables the model's thinking mode, a reserved token `<|reranking|>` (described below), and the opening characters `{"score":`, so that the first generated tokens are the score digits. Decoding is greedy. Both evidence keys are always present, and the model returns an empty list when the page holds no evidence of that kind: it must decide explicitly whether to point at something, rather than simply stop early. The rule for `bbox_2d` asks for one region that contains the visual evidence *and* the context that makes it readable, such as the row, column, axis or legend labels of a value in a table or chart, a definition that is central to our results (Section 6). The rule for `text` asks for every sentence needed to answer the query and no other, since the page text is cut down to exactly this list.

Reranking with the generated score. MMProvence can produce a relevance score in two ways from the same forward pass. We introduce the `<|reranking|>` token, a reserved vocabulary entry whose embedding is learned during training, to obtain a score in the same way as Qwen3-VL-Reranker: a head, initialized from the `yes` and `no` rows of the language-modeling head, reads the hidden state at this token and outputs $\sigma(l_{\text{yes}} - l_{\text{no}})$ (Section 2). The model also writes its score as text, as the first field of the JSON object. Both scores are trained to match those of Qwen3-VL-Reranker-8B (Section 4), and we rank candidates by the textual score, which is parsed together with the evidence and requires no additional head at inference (Appendix D compares the two). A score that cannot be parsed is set to 0, and the sort is stable with respect to the retriever's order, so that a systematic failure degrades to the retriever's ranking rather than to a random one.

Pruning at zero cost. Reranking must process all K candidates, and MMProvence generates the full object for every one of them. Once the ranking is known, the box and the sentences of each of the k surviving pages are applied to its image and text. The pruning stage therefore issues no model call: it only crops images and selects strings. In our implementation, the raw outputs are stored alongside the ranking and read back by the pruning stage, so the pruning variants compared in Section 6 (cropping only, text only, or both) are applied to the same reranking outputs. The cost of pruning is thus moved into the reranking call as additional output tokens: MMProvence-2B generates about 190 tokens per candidate for the median query, most of them the copied sentences.

Answering with sentence indices. Since copying sentences dominates the output, we also train a variant of MMProvence that points at them instead. The page text is shown to the model as a numbered list of sentences, `[1] ... [2] ...`, and the `text` field holds sentence numbers, e.g., `"text": [2, 5]`, rather than copied strings. Everything else is unchanged: the prompt, the score, the box and the geometry. This shortens the `text` field from 62–158 tokens per candidate to 20 or fewer (per-domain medians), and the whole output from about 190 to about 54 tokens. It also removes the matching step: an index identifies its sentence exactly, and the generator still receives the source sentence. The page must be split into sentences exactly as it was during training, since

a single boundary shift renumbers every sentence after it, so we reuse the training-time sentence splitter verbatim. We evaluate this variant in the efficiency study (Section 6).

From box to crop. The box is normalized to page coordinates and padded by 2% of the page on each side, a margin of about one text line that absorbs small localization errors. A padded box thinner than 2% of the page along either axis is treated as invalid. The page image is then cropped to the box and passed to the generator at its native resolution, so that the generator's vision encoder and language model only process the tokens of the crop.

From sentences to text. The page text is split into units: first into lines, then each line into sentences, since the headings, list items and table rows of an extracted page carry no sentence-final punctuation. Each returned sentence is matched to the most similar unit that has not been matched yet, by character-level similarity, and is accepted if the similarity ratio reaches 0.7. The generator then receives the matched *source* units in document order, never the strings the model wrote. MMProvence can therefore only delete content from a page, it cannot rewrite it or add to it, unlike abstractive compressors whose summaries can introduce errors (Xu et al., 2024).

Failing open. The image and the text are resolved independently: an unparseable output, a missing key, a box that fails validation, or a sentence list of which no item matches the page text leaves that modality unchanged, and a failure in one field never affects the other. Only an explicit empty list removes a modality, since it is the model's statement that the page holds no evidence of that kind. We fail open because the two kinds of error are asymmetric: irrelevant content that is kept costs the generator little, whereas relevant content that is deleted cannot be recovered downstream.

Two starting points. Our main model starts from a general-purpose VLM, Qwen3.5 (Qwen Team, 2026), which must learn both to rank and to point at its evidence; we train it at three scales, MMProvence-2B, -4B and -9B. We also explore starting from a purpose-trained reranker, Qwen3-VL-Reranker-2B, as Provence does for text. This model runs as a separate pruning call, so we report it as an exploration of the training recipe rather than as a zero-cost system (Appendix E).

4 TRAINING

We train MMProvence from Qwen3.5 (Qwen Team, 2026) at three scales in two phases: the first teaches the model to rank, and the second to point at its evidence while it keeps ranking.

Objective. The teacher is Qwen3-VL-Reranker-8B, whose relevance probability $P_T = \sigma(l_{\text{yes}} - l_{\text{no}})$ is the score target for every query-candidate pair. The student is trained on the sum of two terms with equal weight: a mean squared error between the output of the score head at the `<|reranking|>` token (Section 3) and P_T, and the cross-entropy of the target JSON object, whose `score` field is P_T written as an integer from 0 to 100. We fine-tune with LoRA (Hu et al., 2021) (rank 64, $\alpha = 128$) on all attention and MLP projections. We train the score head and the `<|reranking|>` embedding in full, and keep the vision encoder frozen. Since every candidate of a query carries its own teacher score, the student learns graded differences between candidates, not only a relevant/irrelevant split.

Phase 1: ranking. The first phase trains the score alone, with the target $\{$`"score":` $s\}$, on $640k$ queries and 6.0M query-candidate pairs from ten subsets. They cover text passages (the MS MARCO and Natural Questions data of Provence (Bajaj et al., 2016; Kwiatkowski et al., 2019; Chirkova et al., 2025)), natural images (OVEN, InfoSeek and WebQA from M-BEIR (Wei et al., 2024)), and document pages (the training sets of ColPali and VisRAG (Faysse et al., 2025; Yu et al., 2025), and the document subsets of Visual-CoT (Shao et al., 2024)). Each query receives a list of up to ten candidates, built from four sources, and every candidate is labelled with the teacher's score rather than with a binary relevance label:

- **Positive:** the labelled answer to the query.
- **Mined negatives:** the hard negatives shipped with the dataset, i.e., the reranked candidates of Provence for text passages and the negative lists of M-BEIR for natural images.

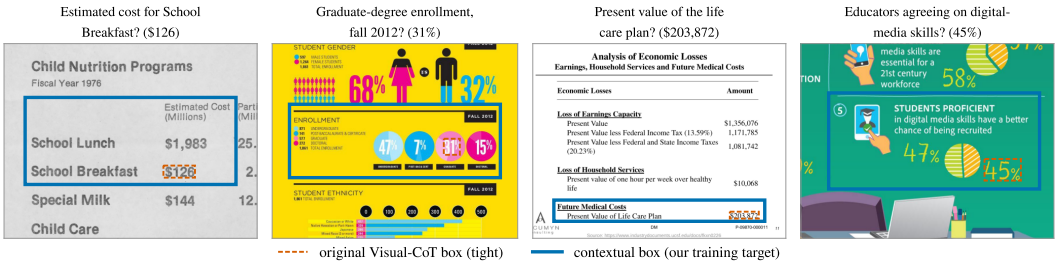

Figure 2: Original Visual-CoT boxes (dashed) and the contextual boxes MMProvence is trained on (solid), on four training examples; the question and its answer are given above each panel. A crop to the original box keeps the value but not what it measures: the column header and row label of a table cell, the legend that says which bubble is *graduate*, the section heading of a line item, or the statement a percentage refers to.

- **Random negatives:** about 1% of the slots hold a random document of the same kind (e.g., another chart for a chart question), so that the student also sees clearly irrelevant candidates, which the other sources rarely provide.
- **Retrieved candidates:** the remaining slots are filled with the top results of our own Qwen3-VL-Embedding-2B index over the subset's corpus, excluding documents identical to the query and known positives. These are the near misses a retriever returns in practice, and exactly what a reranker must learn to order.

Provence ships 50 candidates per query for its text subsets, against 10 to 11 elsewhere; we keep the positive and the nine highest-scoring others, so that text passages, which make up 38% of the queries, do not make up 75% of the pairs. We also keep only the queries whose list contains a ranking to distil: the teacher's best candidate must score at least 0.5 and its second best at least 0.1, computed without the random negatives. We train for one epoch at a learning rate of 10^{-4} and about 260 pairs per step.

Phase 2: evidence. The second phase continues from the phase-1 adapter and trains the full JSON object of Section 3 on 783.6k pairs from 18 subsets. Both evidence keys appear in every target, as an empty list when the document holds no evidence of that kind, so that abstaining is an output the model produces thousands of times rather than an omission it must infer. The subsets provide three kinds of evidence: (i) sentences for text passages (Provence's MS MARCO and Natural Questions annotations, 335.6k pairs), (ii) boxes for images (Visual-CoT, natural images and document pages, 401.6k), and (iii) both for document pages that carry an image and a text (46.4k). All three share one prompt, so that nothing in the input reveals the kind of evidence expected. We train for three epochs at a learning rate of 5×10^{-5}, 336 pairs per step and a maximum length of 8,192 tokens.

Contextual boxes. The boxes of Visual-CoT are drawn tightly around the answer-bearing region, which is precisely the failure mode of cropping: the value survives but the header that says what it measures does not. We therefore re-annotate them (Figure 2). Qwen3.8-27B (Team, 2026) is shown the image, the question and the gold answer, and asked for the rectangle a person would need to see to answer with confidence, including the row and column headers, axis labels or legend of a value and the caption of an object. The training target is the union of this rectangle and the original box, padded by 1% of the image, so the original evidence is contained by construction; 66% of the boxes grow by a factor of two or more (Appendix F). The pages with both kinds of evidence come from two sources. Joint-VisualCoT pairs contextual Visual-CoT boxes on DocVQA, DUDE and SROIE pages with the sentences needed to answer, selected from the page's OCR text after an LLM rewrites it into whole sentences. SynthDoc pages are rendered from text QA corpora with known evidence spans (Chen et al., 2020; Dasigi et al., 2021; Monteiro et al., 2024; Aly et al., 2021), and their boxes are read from the browser layout engine that drew the text, so they are exact (Appendix G).

5 EXPERIMENTAL SETUP

Dataset. We evaluate on the eight public domains of ViDoRe V3 (Loison et al., 2026): computer science, energy, finance (English and French), human resources, industrial maintenance, pharmaceuticals and physics. Each domain is a corpus of 1,110 to 5,244 pages from enterprise document collections, and every page comes with its image and an extracted text transcription. Queries are text-only, come with human-verified answers, and are provided in six languages. We sample 500 queries per domain with a fixed seed, i.e., 4,000 queries in total, the same for every system.

Pipeline and evaluation. We run every experiment with VERGE, our evaluation framework (Appendix B), which follows BERGEN (Rau et al., 2024) and serves every model with vLLM (Kwon et al., 2023). The retriever, Qwen3-VL-Embedding-2B (Li et al., 2026), embeds each page from its image and text jointly and returns $K = 50$ candidates per query. The reranker reads each query together with a candidate's image and text, and the $k = 5$ top-ranked pages are passed to the generator, Qwen3.5-9B (Qwen Team, 2026), as images at their native resolution together with their text. It runs with thinking enabled, greedy decoding and a budget of $40k$ new tokens. We measure answer quality with an LLM judge (LLMeval): Qwen3.5-9B, with thinking enabled, is given the question, the gold answers and the prediction, and outputs 1 if the prediction is correct and 0 otherwise; we report the mean over queries. We measure ranking quality with nDCG@10 against the page-level relevance labels. Compression is reported as the number of generator input tokens, counted with the generator's own processor, and as the shares of text characters and of page area removed.

Baselines. We compare MMProvence with *(i)* no reranking, where the retriever's top-5 pages go straight to the generator; *(ii)* Qwen3-VL-Reranker-2B (Li et al., 2026) without pruning, the full-context baseline and the reranker that MMProvence-2B replaces; *(iii)* Qwen3-VL-Reranker-8B without pruning, our teacher, which is four times larger; and *(iv)* prompted pruners, i.e., Qwen3.5-4B and -9B asked without any training to return the relevant text of each of the top-5 pages, either verbatim (extractive) or as a summary (abstractive), together with one region of its image, with reasoning enabled or disabled. The prompted pruners run after Qwen3-VL-Reranker-2B, with one additional model call per page. To separate the two roles of MMProvence, we also run it as a standalone pruner after Qwen3-VL-Reranker-2B, and we apply its outputs in three ways: cropping the image only, pruning the text only, or both.

Efficiency. We measure latency at batch size 1 on one H100 80GB GPU, with the generator's thinking both enabled and disabled, with the engine of each stage alone on the GPU and the retrieval stage excluded, since it is identical across systems. We time 50 queries per domain, drop the first query of each run as warm-up, and pool the domains with equal weight ($n = 392$ queries). We report medians of the per-query wall-clock time, since a single runaway reasoning trace can take minutes. We complement latency with analytic FLOPs computed from each query's recorded token and image counts, which do not depend on the hardware.

6 EXPERIMENTS

We answer five research questions in the following. Answer quality is the LLMeval score over the 4,000 queries, and ranking quality is nDCG@10 (Section 5). Unless stated otherwise, MMProvence fills both the reranking and the pruning slot of the pipeline, so its pruning stage issues no model call. The alternative recipe that starts from a reranker is evaluated in Appendix E.

RQ1: Does integrating pruning preserve reranking performance? Table 1 compares the ranking of MMProvence, obtained from the score it generates together with its evidence, with that of the Qwen3-VL-Reranker cross-encoders. MMProvence-2B ranks on par with Qwen3-VL-Reranker-2B, the reranker it replaces (nDCG@10 of 0.601 vs. 0.604), and is better on three of the eight domains (Appendix C). MMProvence-4B and -9B rank better than Qwen3-VL-Reranker-2B on all eight domains (0.627 and 0.636), and MMProvence-9B comes within 0.006 of its teacher, Qwen3-VL-Reranker-8B. Ranking quality carries over to the answers: without pruning, LLMeval rises from 0.761 without reranking to 0.827 with Qwen3-VL-Reranker-2B and 0.848 with the 8B version.

Table 1: Ranking and pruning (RQ1–RQ3). The Qwen3-VL-Reranker-2B row, without pruning, is the full-context baseline; Qwen3-VL-Reranker-8B is the teacher of MMProvence. The prompted pruners are Qwen3.5-4B and -9B, with one additional model call per page; *reasoning enabled. Tokens: median generator input tokens per query over the 392 queries of the efficiency study (–: not measured); Red.: their reduction. Text, Image: share of text characters and of page area removed. Δ: LLMeval difference to the baseline. Per-domain nDCG@10 in Appendix C.

Reranker	Pruning	nDCG@10	Tokens	Red.	Text	Image	LLMeval	Δ
None (retriever only)	none	0.496	–	–	0%	0%	0.761	−0.067
Qwen3-VL-Reranker-2B	none	0.604	21.6k	–	0%	0%	0.827	–
Qwen3-VL-Reranker-8B	none	0.642	–	–	0%	0%	0.848	+0.021
MMProvence-2B	crop only	0.601	7.9k	−63%	0%	74.6%	0.822	−0.005
	text only		19.7k	−9%	63.4%	0%	0.825	−0.003
	crop + text		5.3k	−75%	63.4%	74.6%	0.803	−0.024
MMProvence-4B	crop only	0.627	7.8k	−64%	0%	75.0%	0.840	+0.013
	text only		19.4k	−10%	73.0%	0%	0.832	+0.005
	crop + text		4.9k	−77%	73.0%	75.0%	0.811	−0.017
MMProvence-9B	crop + text	0.636	5.3k	−76%	66.0%	74.4%	0.822	−0.006
Qwen3-VL-Reranker-2B + prompted pruner	extractive 4B	0.604	5.6k	−74%	66.2%	69.2%	0.806	−0.021
	extractive 4B*		8.4k	−61%	60.3%	57.9%	0.808	−0.019
	extractive 9B		8.9k	−59%	71.8%	52.3%	0.808	−0.019
	extractive 9B*		8.9k	−59%	56.3%	57.0%	0.821	−0.007
	abstractive 4B		18.8k	−13%	45.3%	10.6%	0.809	−0.019
	abstractive 4B*		15.6k	−28%	49.7%	21.9%	0.814	−0.014
	abstractive 9B		13.0k	−40%	47.2%	35.5%	0.806	−0.022
	abstractive 9B*		14.2k	−34%	49.1%	29.4%	0.814	−0.013

RQ2: Does MMProvence enable context compression with minimal performance loss? Table 1 applies MMProvence's crop, its text pruning, or both. Our conclusions are as follows: *(1)* Since page images account for most of the input, the crop delivers most of the compression: cropping alone keeps about a quarter of the page area and removes 63% of the generator's input tokens, at an LLMeval of 0.822 for MMProvence-2B against 0.827 for the full-context baseline. *(2)* Text pruning removes 63–73% of the text characters, but only 9–10% of the input tokens, for a similarly small change in LLMeval (0.825). *(3)* Applying both reduces the input by 75–77%, from 21.6k to about 5k tokens, and costs more than either alone: MMProvence-2B reaches 0.803 against 0.827. *(4)* Larger models prune at a lower cost, partly because they also rank better (RQ1): MMProvence-4B scores above the baseline with either kind of pruning alone (0.840 and 0.832), and MMProvence-9B keeps an LLMeval of 0.822 while removing 76% of the input.

RQ3: How does MMProvence compare against prompted pruners? The prompted pruners of Table 1 (last block) are Qwen3.5 models asked, without any training, to return the relevant text of a page and one region of its image (Section 5). Unlike MMProvence, they are separate models and are not zero-cost: each runs after the reranker and issues one additional call to a 4B or 9B VLM for every page that reaches the generator, which with reasoning enabled includes a reasoning trace per page. The best prompted pruner, extractive Qwen3.5-9B with reasoning, reaches an LLMeval of 0.821 while reducing the input by 59%; MMProvence-9B reaches 0.822 at a 76% reduction, and without any additional call. At a similar compression, the extractive Qwen3.5-4B without reasoning (-74%) scores 0.806, between MMProvence-2B (0.803) and MMProvence-4B (0.811, -77%). The abstractive pruners compress the least (-13 to -40%), since their crops keep 65–89% of the page, and do not answer better than the extractive ones. Finally, the prompted pruners delete the whole page image on 7–22% of the pages, which MMProvence never does.

RQ4: Which design choices matter? We run three ablations. *Box target.* Two earlier 4B versions of MMProvence, trained with one evidence key per example and differing chiefly in their box targets, isolate the definition of the box. Added on top of text pruning, crops to tight boxes around the answer-bearing region lower LLMeval from 0.819 to 0.686 (-0.133), whereas crops to contextual boxes (Section 4) lower it from 0.833 to 0.806 (-0.027). Applied alone, the contextual crops of five

Table 2: Efficiency (RQ5): median seconds per query at batch size 1, with the generator's thinking enabled (Think) or disabled (No think), and median analytic TFLOPs per query (thinking enabled), over $n = 392$ queries. The first row is the full-context pipeline; MMProvence's pruning stage adds 0.16–0.17 s.

		Qwen3.5-9B generator			Qwen3.8-27B generator		
Reranker	Rerank	Think	No think	TFLOPs	Think	No think	TFLOPs
Qwen3-VL-Reranker-2B	6.61	21.80	10.94	1,180	35.06	14.99	2,056
MMProvence-2B	7.37	19.37	9.48	818	32.65	12.38	1,060
MMProvence-4B	8.09	20.03	10.45	1,428	33.41	13.61	1,648
MMProvence-9B	12.08	24.31	14.21	2,484	37.99	17.30	2,732
MMProvence-2B, sentence indices	5.22	16.86	7.37	789	30.80	10.62	989

checkpoints change LLMeval by -0.002 to -0.007 while keeping about a quarter of the page area. *Pruning inside reranking.* Running the same checkpoints as standalone pruners behind Qwen3-VL-Reranker-2B separates their pruning from their ranking. With crop + text, MMProvence-2B, -4B and -9B reach an LLMeval of 0.802, 0.803 and 0.810 as standalone pruners, against 0.803, 0.811 and 0.822 in both slots: at 2B, where the two rankings tie (nDCG@10 of 0.604 vs. 0.601), both arrangements score alike, and at 4B and 9B the joint one is better, in line with its better ranking. Generating the evidence at reranking depth, for all 50 candidates rather than for the five surviving pages, therefore costs no answer quality. *Phase 1.* Training MMProvence-2B on phase 2 alone, with the same data and hyperparameters but without the phase-1 warm start, lowers nDCG@10 from 0.601 to 0.594, below Qwen3-VL-Reranker-2B, and LLMeval from 0.803 to 0.791, although this variant trains for 21% more phase-2 steps.

RQ5: Is MMProvence more efficient than the full-context pipeline? Table 2 reports latency and compute per query (Section 5). *Generation.* Pruning removes most of the generator's work before decoding: with Qwen3.5-9B, preprocessing and prefill take about 0.5 s per query instead of 3.0 s, and the compute spent on the input falls from 537 to 99 TFLOPs (-82%); the vision encoder's compute falls by 89%, since its attention is quadratic in the patches of each image. Decoding, however, runs at 7.0–7.3 ms per output token in every system and, with thinking enabled, takes 12.6 of the 15.5 s of full-context generation. *Reranking.* The cost of pruning is paid in reranking instead: MMProvence generates 140 to 190 tokens per candidate instead of one, and reranking takes 0.75, 1.48 and 5.47 s longer at 2B, 4B and 9B. *Overall.* MMProvence-2B is faster than the full-context pipeline by 11% with thinking and 13% without, and uses 31% fewer TFLOPs. MMProvence-4B is slightly faster but uses 21% more compute, and MMProvence-9B is slower and uses twice the compute. With Qwen3.8-27B, pruning removes 79–80% of a costlier input side, so MMProvence-2B and -4B save 48% and 20% of the compute; their latency savings grow to 17% and 9% without thinking and shrink to 7% and 5% with it, as the 27B's decoding dominates.

Sentence indices. The variant that answers with sentence numbers (Section 3) cuts MMProvence-2B's reranking output from 9,578 to 2,700 tokens per query (-72%), so its reranking takes 5.22 s, less than Qwen3-VL-Reranker-2B (6.61 s), at a slightly higher compute (673 vs. 609 TFLOPs). It also leaves the generator fewer input tokens ($4.2k$ against $5.3k$), so the whole pipeline takes 23% and 33% less time than with the full context, with and without thinking.

7 Conclusion

In this work, we present MMProvence, a VLM reranker that generates, for each query–page pair, a relevance score, a box on the page image and the page sentences that answer the query, in a single call, so that multimodal RAG contexts are pruned without any additional model call. On the eight domains of ViDoRe V3, MMProvence-2B ranks on par with the cross-encoder it replaces and cuts the generator's median input from $21.6k$ to $5.3k$ tokens, at an LLMeval of 0.803 against 0.827, and MMProvence-4B and -9B rank better than this cross-encoder. What the box is trained to cover decides the cost of cropping: boxes that include the context that makes the evidence readable make cropping nearly free, whereas boxes around the smallest answer-bearing region do not. We will release our code, checkpoints and training data; Appendix A discusses limitations and future work.

AI USE STATEMENT

We used an AI coding assistant (Claude Code) to implement parts of the evaluation framework and the analysis scripts, to launch and monitor experiments, to compute the reported statistics from the experiment outputs, and to draft and edit the text of the paper from our results and notes. The research questions, the method and the design of the experiments are our own. Every number in the paper is produced by scripts that read the experiment outputs, and the authors checked the numbers, the claims and the code against these outputs. Every bibliography entry was checked against its source. We have reviewed all AI-assisted work and take responsibility for the final content of this paper, including text, claims and artifacts produced with the aid of generative AI.

REFERENCES

Rami Aly, Zhijiang Guo, Michael Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. Feverous: Fact extraction and verification over unstructured and structured information. *arXiv preprint arXiv:2106.05707*, 2021.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pp. 19–35. Springer, 2024.

Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Wang, and William W Cohen. Open question answering over tables and text. *arXiv preprint arXiv:2010.10439*, 2020.

Nadezhda Chirkova, Thibault Formal, Vassilina Nikoulina, and Stéphane Clinchant. Provence: efficient and robust context pruning for retrieval-augmented generation. In *International Conference on Learning Representations*, volume 2025, pp. 38082–38106, 2025.

Jaemin Cho, Debanjan Mahata, Ozan Irsoy, Yujie He, and Mohit Bansal. M3docrag: Multi-modal retrieval is what you need for multi-page multi-document understanding. *arXiv preprint arXiv:2411.04952*, 2024.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4599–4610, 2021.

Manuel Faysse, Hugues Sibille, Tony Wu, Bilel Omrani, Gautier Viaud, Céline Hudelot, and Pierre Colombo. Colpali: Efficient document retrieval with vision language models. In *International Conference on Learning Representations*, volume 2025, pp. 61424–61449, 2025.

Robert Friel, Masha Belyi, and Atindriyo Sanyal. Ragbench: Explainable benchmark for retrieval-augmented generation systems. *arXiv preprint arXiv:2407.11005*, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Taeho Hwang, Soyeong Jeong, Sukmin Cho, SeungYoon Han, and Jong C Park. Dslr: Document refinement with sentence-level re-ranking and reconstruction to enhance retrieval-augmented generation. In *Proceedings of the 3rd Workshop on Knowledge Augmented Methods for NLP*, pp. 73–92, 2024.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 13358–13376, 2023.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

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

Mingxin Li, Yanzhao Zhang, Dingkun Long, Keqin Chen, Sibo Song, Shuai Bai, Zhibo Yang, Pengjun Xie, An Yang, Dayiheng Liu, et al. Qwen3-vl-embedding and qwen3-vl-reranker: A unified framework for state-of-the-art multimodal retrieval and ranking. *arXiv preprint arXiv:2601.04720*, 2026.

António Loison, Quentin Macé, Antoine Edy, Victor Xing, Tom Balough, Gabriel de Souza P Moreira, Bo Liu, Manuel Faysse, Céline Hudelot, and Gautier Viaud. Vidore v3: A comprehensive evaluation of retrieval augmented generation in complex real-world scenarios. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16570–16600, 2026.

Xueguang Ma, Sheng-Chieh Lin, Minghan Li, Wenhu Chen, and Jimmy Lin. Unifying multimodal retrieval via document screenshot embedding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 6492–6505, 2024.

Youssef Mohamed, Mohamed Elhoseiny, Thibault Formal, and Nadezhda Chirkova. Xprovence: Zero-cost multilingual context pruning for retrieval-augmented generation. In *European Conference on Information Retrieval*, pp. 470–480. Springer, 2026.

Joao Monteiro, Pierre-Andre Noel, Etienne Marcotte, Sai Rajeswar, Valentina Zantedeschi, David Vazquez, Nicolas Chapados, Christopher Pal, and Perouz Taslakian. Repliqa: A question-answering dataset for benchmarking llms on unseen reference content. *Advances in Neural Information Processing Systems*, 37:24242–24276, 2024.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, et al. Large language models are effective text rankers with pairwise ranking prompting. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 1504–1518, 2024.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

David Rau, Hervé Déjean, Nadezhda Chirkova, Thibault Formal, Shuai Wang, Stéphane Clinchant, and Vassilina Nikoulina. Bergen: A benchmarking library for retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 7640–7663, 2024.

Hao Shao, Shengju Qian, Han Xiao, Guanglu Song, Zhuofan Zong, Letian Wang, Yu Liu, and Hongsheng Li. Visual cot: Advancing multi-modal language models with a comprehensive dataset and benchmark for chain-of-thought reasoning. *Advances in Neural Information Processing Systems*, 37:8612–8642, 2024.

Haitian Sun, William Cohen, and Ruslan Salakhutdinov. Conditionalqa: A complex reading comprehension dataset with conditional answers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3627–3637, 2022.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 14918–14937, 2023.

Qwen Team. Qwen3. 8-max: A new bar for coding and cowork. *Qwen release announcement*, 2026.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Christophe Van Gysel and Maarten de Rijke. Pytrec_eval: An extremely fast python interface to trec_eval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 873–876, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Cong Wei, Yang Chen, Haonan Chen, Hexiang Hu, Ge Zhang, Jie Fu, Alan Ritter, and Wenhu Chen. Uniir: Training and benchmarking universal multimodal information retrievers. In *European Conference on Computer Vision*, pp. 387–404. Springer, 2024.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *International Conference on Learning Representations*, volume 2024, pp. 43478–43502, 2024.

Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL `https://github.com/hydra-ecosystem/hydra`.

Shi Yu, Chaoyue Tang, Bokai Xu, Junbo Cui, Junhao Ran, Yukun Yan, Zhenghao Liu, Shuo Wang, Xu Han, Zhiyuan Liu, et al. Visrag: Vision-based retrieval-augmented generation on multi-modality documents. In *International Conference on Learning Representations*, volume 2025, pp. 21074–21098, 2025.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. Sparsevlm: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*, 2024.

Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Bendersky. Beyond yes and no: Improving zero-shot llm rankers via scoring fine-grained relevance labels. In *Proceedings of the 2024 conference of the North American chapter of the Association for Computational Linguistics: Human language technologies (volume 2: short papers)*, pp. 358–370, 2024.

A Limitations and future work

Our study has several limitations. *(i)* We evaluate answer quality on one benchmark, whose queries are text-only, with one generator and one LLM judge, which does not see the page images; the larger Qwen3.8-27B generator is evaluated for efficiency only. *(ii)* Pruning both modalities costs more than pruning either alone, e.g., 0.024 LLMeval for MMProvence-2B against at most 0.005 for the crop or the text alone. Moreover, MMProvence makes discrete pruning decisions and offers no threshold that trades compression for quality, as the per-token probabilities of Provence do. *(iii)* Pruning shortens everything before decoding but not the decoding itself, which dominates latency when the generator reasons, and generating the evidence makes reranking slower, so that with MMProvence-9B the pipeline is slower than with the full context. *(iv)* MMProvence predicts one box per page, so evidence spread over distant regions of a page can only be covered by a single rectangle, and each page is pruned independently of the other retrieved pages. *(v)* ViDoRe V3 has no box annotations, so the localization quality of the boxes is measured only through its effect on the answers. Future work includes combining MMProvence with visual token reduction inside the generator (Section 2), predicting several boxes per page, and exposing a pruning strength that can be adjusted at inference.

B VERGE: VISUAL END-TO-END RAG EVALUATION

All experiments in this paper are run with VERGE, an evaluation framework for end-to-end multimodal RAG that we implemented and will release. It is the multimodal counterpart of BERGEN (Rau et al., 2024), whose design it follows, so that a researcher familiar with BERGEN can use it directly. This section describes its design and the features that make large-scale evaluation practical.

Design philosophy. A run is fully described by its configuration. Each component type has a folder of Hydra (Yadan, 2019) configuration files, and a configuration names the class that implements the component together with its constructor arguments; the pipeline instantiates it at run time. Every component type is defined by an abstract interface (Table 3), and the pipeline only calls a model through that interface. A new retriever, reranker, pruner, generator, judge or dataset is therefore added by implementing one class and writing one configuration file, and every experiment in this paper is a command line that selects components, e.g., `retriever=qwen3-vl-embedding-2b reranker=qwen3-vl-reranker-2b generator=qwen3.5-9b dataset=vidore_v3_energy`. A single orchestrator runs the stages in order, i.e., retrieval, reranking, pruning, generation and evaluation, and writes each experiment to its own folder with its configuration, predictions and metrics.

Table 3: Component types of VERGE, the interface each one implements, and the instances used in this paper.

Component	Interface	Instances in this paper
Dataset	processor returning query and page rows	ViDoRe V3, eight domains
Retriever	encode queries and pages; similarity	Qwen3-VL-Embedding-2B
Reranker	score the candidates of a query	Qwen3-VL-Reranker-2B and -8B; MMProvence
Pruner	prune each surviving page, one output per page	MMProvence; prompted extractive and abstractive pruners; the recipe of Appendix E
Generator	answer from the query and the pages	Qwen3.5-9B; Qwen3.8-27B
Evaluator	score a prediction against the gold answers	Qwen3.5-9B judge (LLMeval)

Multimodal data contract. Every query and page is a row with an identifier, a text field, and an image field, which holds the path to one or several images; queries also carry their gold answers. Either field may be empty, e.g., for text-only queries or image-only pages. Images are referenced by path and loaded only when a model needs them, so that a corpus of thousands of page images never has to be held in memory. Retrievers, rerankers and pruners can encode a page from its image, its text, or both jointly, and a retriever declares the modalities it consumes on each side, so that pairing it with a dataset that lacks one fails before any page is encoded.

Pruning as a pipeline stage. Compared with BERGEN, the pipeline has an additional pruning stage between reranking and generation. A pruner returns exactly one output per surviving page, so that pruning can shrink the content of a page but never change which pages reach the generator, and a malformed output leaves the corresponding modality of the page unchanged (Section 3). A single component can fill both the reranking and the pruning slot, as MMProvence does: during reranking, it writes its raw outputs to a file stored next to the ranking, and the pruning stage replays this file without loading a model. The pruning variants of Section 6, i.e., cropping only, pruning the text only, or both, are thus applied to the same reranking outputs at no additional GPU cost.

Performance. Several features keep full-scale evaluation tractable. *(i)* Every model is served with vLLM (Kwon et al., 2023), including the embedding model and the cross-encoder rerankers, whose yes/no scores are read from the log-probabilities of a single generated token. *(ii)* The output of every stage is cached on disk: the processed dataset, the page and query embeddings, the retrieval and reranking runs in the TREC format, and the pruned contexts. Cache files are named after the component, the upstream stage and the depth they depend on, so that a change upstream never

silently reuses a stale result, while an ablation of a later stage reuses all earlier computation; a finished experiment is detected and not recomputed. *(iii)* Models are loaded lazily, only for the stages that are not cached, and each stage's engine can be released before the next one is loaded, so that the pipeline fits on a single GPU regardless of the number of components. *(iv)* The LLM judge runs data-parallel, with one engine per free GPU, and rows are distributed across engines in an interleaved order that balances long and short predictions. Judging 5,196 predictions takes about 3 minutes on 8 GPUs against about 15 minutes on one. Finished scores are checkpointed, so an interrupted evaluation resumes where it stopped. *(v)* Page images can be handed to the engines as decoded images rather than as encoded strings, which we do in the latency study: for Qwen3-VL-Reranker-2B, this alone halves the reranking time of a 50-candidate query, from 16.2 to 7.5 s, with identical scores.

Metrics and measurements. VERGE reports ranking metrics (nDCG@k, Recall@k and MRR, computed with pytrec_eval (Van Gysel & de Rijke, 2018)), answer metrics (the LLM judge, as well as match, exact match and F1 against the gold answers), and compression metrics for pruning (text characters removed, page area retained, and images dropped). Every run also records, for each stage, whether it was read from the cache and its wall-clock time, together with the per-query prompt and generated token counts of the reranker and the generator. An optional mode generates one query at a time and records the engine's time to first token, prefill and decoding times; the latency study of Section 6 uses it, and the analytic FLOPs are computed from the recorded token and image counts.

C Ranking quality per domain

Table 4 breaks down the nDCG@10 of Table 1 by domain. MMProvence-2B is better than Qwen3-VL-Reranker-2B on three of the eight domains, and MMProvence-4B and -9B are better on all eight.

Table 4: Ranking quality per domain: nDCG@10 per domain (computer science, energy, finance in English and French, human resources, industrial maintenance, pharmaceuticals, physics), and LLMeval when the reranked pages are passed to the generator without pruning. The first row passes the retriever's top-5 pages to the generator directly. †Teacher of MMProvence.

	nDCG@10									LLMeval
Reranker	CS	En	FinE	FinF	HR	Ind	Pha	Phy	Avg.	
None (retriever only)	0.718	0.549	0.518	0.293	0.467	0.404	0.558	0.465	0.496	0.761
Qwen3-VL-Reranker-2B	0.785	0.665	0.664	0.442	0.593	0.529	0.636	0.515	0.604	0.827
Qwen3-VL-Reranker-8B†	0.820	0.703	0.715	0.471	0.648	0.574	0.679	0.526	0.642	0.848
MMProvence-2B	0.776	0.671	0.657	0.436	0.600	0.522	0.641	0.507	0.601	0.833
MMProvence-4B	0.801	0.693	0.689	0.458	0.621	0.556	0.669	0.530	0.627	0.842
MMProvence-9B	0.808	0.696	0.706	0.467	0.639	0.561	0.679	0.532	0.636	–

D Score head vs. textual score

MMProvence produces two relevance scores in the same forward pass (Section 3): the output of the score head at the <|reranking|> token, and the integer it writes as the first field of its JSON object. Both are trained to match the relevance probability of Qwen3-VL-Reranker-8B (Section 4). Figure 3 compares them over the phase-2 checkpoints of MMProvence-4B, as the mean squared error against the teacher's probability on 1,043 held-out validation pairs (up to 64 per validation subset), with the textual score divided by 100. The score head is the more accurate of the two and improves steadily, down to 0.0032, while the error of the textual score fluctuates from one checkpoint to the next and reaches 0.0042 at the end of training; at the checkpoint we use, the two errors are 0.0033 and 0.0052.

In absolute terms, both errors are small: an MSE of 0.0052 is a root-mean-square error of 0.072 on the $[0, 1]$ scale of the teacher, i.e., about 7 points on the 0–100 scale of the textual score, against about 6 points for the score head. Reranking only uses the order of the candidates, so a difference of

this size leaves little room to change the results of the RAG pipeline, and the textual score already ranks better than Qwen3-VL-Reranker-2B (Section 6). We therefore rank candidates by the textual score, which is the more straightforward implementation: it is parsed from the generated output together with the evidence, and requires no additional head at inference.

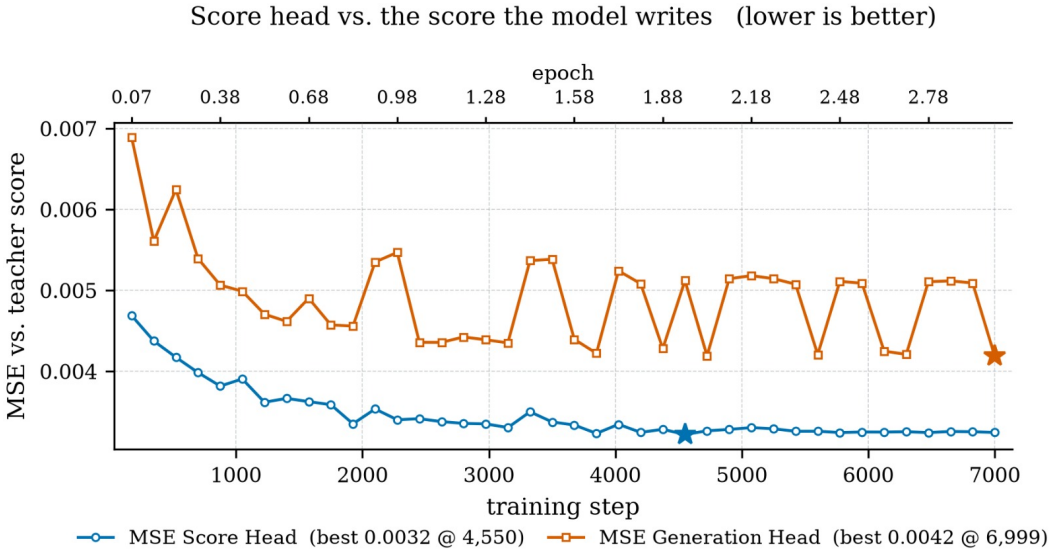

Figure 3: Mean squared error against the teacher's relevance probability of the score head at the <|reranking|> token (blue) and of the score MMProvence writes in its output (orange, divided by 100), over the phase-2 checkpoints of MMProvence-4B, on 1,043 held-out validation pairs. Stars mark the best checkpoint of each score.

E STARTING FROM A RERANKER

As an alternative to training MMProvence from a general-purpose VLM, we explore the route that Provence takes for text: starting from a purpose-trained reranker and adding pruning (Section 3).

Training. This recipe fine-tunes Qwen3-VL-Reranker-2B with LoRA (rank 64, $\alpha = 128$), updating its language model and multimodal projector. Since this model already ranks, it is trained only to point at evidence, with the output {"bboxes_2d": [...], "selected_sentences": [...]}. The first stage trains two single-modality tasks: boxes on Visual-CoT's DocVQA, DUDE and SROIE pages, with the contextual boxes of Section 4, and sentence selection on the Provence annotations. The second stage adds the joint task on Joint-VisualCoT (Section 4), mixed with the two single-modality tasks at a 3:1:1 ratio, together with negatives in which the evidence sentences are deleted or the page image is replaced (one negative per five positives).

Results. We evaluate the final model of this recipe. Since its output holds no relevance score, it runs as a separate pruning call behind the unmodified Qwen3-VL-Reranker-2B, one call per page that reaches the generator. Its ranking is therefore that of the baseline, and its pruning is not zero-cost.

Table 5 shows the results. The text pruning of this model is the most aggressive of all our models: it removes 80% of the text characters, against 63% for MMProvence-2B, for an LLMeval of 0.819 against 0.827 without pruning. Its crop, however, lowers LLMeval by a further 0.047, to 0.772, whereas the crop of MMProvence-2B costs 0.022 on top of its text pruning. Both keep a similar share of the page, 23.0% and 25.4%: at a similar retained area, where the box is placed matters as much as how large it is. Starting from a reranker gives the model a ranking it does not need to learn, but the pruning stage then requires a model call of its own, which is what MMProvence avoids.

Table 5: The recipe that starts from a reranker (Qwen3-VL-Reranker-2B + LoRA pruner), run as a separate pruning call behind Qwen3-VL-Reranker-2B, compared with MMProvence-2B. Columns as in Table 1.

Pruner	Pruning	Tokens	Red.	Text	Image	LLMeval	Δ
Qwen3-VL-Reranker-2B	none	21.6k	–	0%	0%	0.827	–
Qwen3-VL-Reranker-2B	text only	18.9k	−13%	79.9%	0%	0.819	−0.009
+ LoRA pruner	crop + text	3.7k	−83%	80.0%	77.0%	0.772	−0.056
MMProvence-2B	text only	19.7k	−9%	63.4%	0%	0.825	−0.003
	crop + text	5.3k	−75%	63.4%	74.6%	0.803	−0.024

F CONTEXTUALIZED VISUAL-COT

Visual-CoT (Shao et al., 2024) pairs 434,265 questions over images from twelve sources with a box around the region the answer is read from. Its boxes are drawn tightly around the answer: they cover 14.6% of the image on average, and a crop to them frequently removes what is needed to interpret the answer, such as the header of a table cell or the object that a price tag belongs to (Figure 2). We re-annotate every row with a contextual box and will release the result as Contextualized Visual-CoT. It keeps the rows, image files, keys and order of Visual-CoT and changes only the box; the original box and the raw box of the re-annotating model are stored alongside, so the change is reversible. Each row also carries the relevance score that Qwen3-VL-Reranker-8B assigns to its question and image, which we use as the distillation target (Section 4).

Re-annotation. Qwen3.8-27B (Team, 2026) is shown the image, the question and the gold answer, and asked for the rectangle a reader would need to see to answer with confidence, including the context that makes the evidence interpretable (Figure 4). It decodes greedily with thinking enabled, one image per prompt, and returns a box on a 1000×1000 grid. The released box is the smallest rectangle containing both the model's box and the original one, padded by 1% of the image on each side. By construction, it contains everything the original box contained, so switching to the contextual boxes cannot lose annotated evidence. Rows the model fails to answer keep their original box. Re-annotating all rows took about six hours on eight H100 GPUs.

```
system: You are shown an image, a question about it, and the correct answer. Return the
axis-aligned rectangle a person would have to see in order to answer the question with
confidence.
The rectangle must contain ALL of the visual evidence the answer depends on -- if the
answer depends on several things, it must enclose every one of them.
It must ALSO contain the surrounding context that makes that evidence interpretable,
because evidence alone is often ambiguous:
- for a value in a table or chart, include the row and column headers, axis labels, or
legend entry that say what the value means;
- for an object, include the label, caption, or title that names it;
- for a comparison or a count, include every item being compared or counted;
- for a relation between things, include both things and the space between them.
Prefer a slightly larger rectangle over one that cuts off context. Do not return the whole
image unless the answer genuinely depends on all of it.
Coordinates are integers on a 0-1000 grid, left to right and top to bottom. The grid is
relative to the image, so 500 is the middle of the image on that axis no matter how tall or
wide the image is.
Reply with JSON and nothing else: {"bbox_2d": [x1, y1, x2, y2]}
user: (image) Question: {question}
Answer: {answer}
```

Figure 4: The prompt with which Qwen3.8-27B re-annotates the boxes of Visual-CoT.

Statistics. Table 6 reports the box area before and after re-annotation. The model returns a strictly larger region for 99.8% of the rows. The mean box grows from 14.6% to 34.6% of the image, i.e., by a factor of 2.37, of which the padding alone contributes a factor of 1.04, and 65.9% of the boxes at least double in area. The growth is largest where the original boxes are tightest, i.e., for text in natural images (TextCaps and TextVQA) and for document pages (DUDE, InfographicsVQA and DocVQA), and smallest for CUB, whose boxes already cover over a third of the image. Across all

rows, 18.3% of the boxes grow by less than 25%, while 17.9% grow by a factor of 20 or more, e.g., when a single value or word is extended to the row, column or sign that names it.

Table 6: Contextualized Visual-CoT: rows per subset, mean box area as a share of the image with the original and the contextual boxes, mean growth factor, and share of boxes whose area at least doubles.

Subset	Rows	Original area	Contextual area	Growth	$\geq 2\times$
CUB	10,056	35.0%	38.8%	1.11×	0.5%
DocVQA	33,453	5.4%	10.6%	1.96×	80.8%
DUDE	11,735	2.7%	11.7%	4.40×	89.7%
Flickr30k	135,735	14.4%	46.2%	3.21×	70.1%
GQA	98,149	18.0%	35.1%	1.95×	56.5%
InfographicsVQA	15,055	8.8%	22.7%	2.58×	81.4%
Open Images	43,053	21.9%	40.4%	1.84×	53.6%
SROIE	2,486	9.3%	15.0%	1.61×	68.2%
TextCaps	32,152	4.3%	17.1%	3.97×	82.6%
TextVQA	18,524	4.4%	15.1%	3.43×	80.0%
Visual7W	30,491	19.7%	41.0%	2.08×	58.3%
VSR	3,376	36.8%	66.2%	1.80×	47.2%
All	434,265	14.6%	34.6%	2.37×	65.9%

Use in training. The twelve subsets provide the $401.6k$ box pairs of phase 2 after evaluation rows are held out, and the DocVQA, DUDE and SROIE pages of Joint-VisualCoT use the same contextual boxes (Section 4).

G SynthDoc

Phase 2 needs document pages that carry both kinds of evidence, a box and the sentences inside it. Human-annotated document datasets rarely provide both, and boxes derived from OCR are approximate. SynthDoc instead renders text QA corpora whose evidence is known into page images, and reads the box out of the layout engine that drew the text, so that the box is a measurement rather than an annotation. We will release it with the other training data.

Sources. Four corpora mark the evidence of each answer, and each contributes a different document shape: OTT-QA (Chen et al., 2020), a Wikipedia table with its linked passages, whose evidence is the passage or table row that grounds the answer; RepLiQA (Monteiro et al., 2024), reference articles about invented entities, which cannot be answered from a model's parametric memory, with the annotated answer sentence as evidence; FEVEROUS (Aly et al., 2021), Wikipedia pages that verify a claim, whose evidence is the cited sentences, list items and table cells, and whose query is the claim and answer its label (claims without enough information are excluded); and Qasper (Dasigi et al., 2021), NLP research papers with the sentences that annotators cited as evidence. Queries and answers are carried over verbatim. We keep only documents whose evidence spans at least 600 characters, so that the box covers a real region of the page. Three other corpora were tried and rejected: in MuSiQue (Trivedi et al., 2022), multi-hop evidence lies in distant paragraphs, so a single box would enclose the distractors between them; ConditionalQA (Sun et al., 2022) has too few documents with a long enough evidence span; and RAGBench (Friel et al., 2024) stores its financial tables as serialized lists, which render as source code rather than as a table.

Rendering. Each document is laid out in HTML and CSS and captured with headless Chromium. Before the capture, every evidence word is measured with the DOM's `Range.getClientRects()`, which returns the rectangles the browser used to paint its glyphs, and the union of these rectangles, padded by 6 CSS pixels on each side, is the evidence box. The style is sampled per render: 6 page geometries (A4, Letter, slide, poster, infographic and wide), 8 layouts (e.g., running text, cards, bands or tiles), 16 color themes, 13 open-licence typefaces, 4 table styles, one or two columns, and continuous line height, margins and type size, the latter solved so that the text fills a target share of the page. Decorations that are not content, such as section numbers,

Table 7: SynthDoc pages per source corpus and split. Splits are disjoint on the source document.

Source corpus	Train	Validation	Test
OTT-QA	1,435	63	63
Qasper	1,434	62	62
RepLiQA	1,434	62	62
FEVEROUS	697	63	63
Pages (source documents)	5,000 (2,283)	250 (165)	250 (203)

bullets and rules, are drawn by CSS rather than as text, so that they never enter the page text or the box.

Quality gates. A render is kept only if it passes twelve automatic checks, with no human review. The main ones require that the box covers 20–75% of the page; that at least 85% of the text inside the box is evidence (the box is the union of the evidence, so its recall is 1 by construction and only its precision can fail); that the evidence is at most 55% of the page text (70% for FEVEROUS, whose short pages are mostly evidence); that at least 55% of the usable page carries content; that the sampled typeface was loaded and nothing overflowed the page; and that the text recovered from the measured boxes equals the source evidence character for character. About one render in four passes.

Statistics. SynthDoc has 5,500 pages, split by source document so that no document appears in two splits (Table 7). The evidence makes up a median 47% of a page's text, and the box covers a median 25.5% of the page; a constant box, predicted identically for every page, already reaches an IoU of 0.532 with the gold boxes, which is the baseline a localization model should be compared against on this data. Every page is a positive for its query, yet the scores of Qwen3-VL-Reranker-8B, given the page image and text, spread over the whole range (median 0.578 on the training split): they are soft targets that are low where the evidence is buried in a dense layout or phrased far from the query. Each row stores the query, the answer, the page text and image, the box, the evidence sentences, the teacher score, and the full rendering settings.

Use in training. The 5,000 training pages form one of the two sources of pages with both kinds of evidence in phase 2, together with Joint-VisualCoT (Section 4). Their target holds the teacher score, the box on the 1000×1000 grid, and the evidence sentences.

H PROMPTS

This section gives the prompts of the models in our pipeline verbatim. Text in braces is a placeholder that is filled for each query or page, and *(page image)* marks where the page image is inserted.

MMProvence. Figure 5 gives the full prompt of MMProvence; it is the prompt used during training. Since the assistant turn is prefilled (Section 3), the prompt is passed to the model as a rendered string rather than through its chat template. The query is shown as text only, and the page image precedes the page text in the document slot. The first sentence is the instruction of Qwen3-VL-Reranker, which refers to an `Instruct` field; the user turn of MMProvence has none, and we keep the wording the model was trained with. The sentence-index variant of Section 3 shows the page text as a numbered list of sentences and changes only the schema line and the rule for `text` (Figure 6).

Prompted pruners. The extractive and abstractive pruners of Table 1 share one prompt structure and differ in their system turn, their instruction and the tag that holds the text (Figure 7). The query is shown as text only, and the page text is truncated to its first 2,000 words. Reasoning is enabled or disabled through the switch of the chat template, with the prompt unchanged; when it is enabled, the tags are read after the reasoning trace. The region is either `FULL`, which keeps the page image, `NONE`, which deletes it, or a box, to which the image is cropped. As for MMProvence, a missing or malformed tag leaves the corresponding modality of the page unchanged.

system: Judge whether the Document meets the requirements based on the Query and the Instruct provided, and point to the evidence you used.
Reply with one JSON object and nothing else -- no explanation, no code fence:
{"score": <integer>, "bbox_2d": [x0, y0, x1, y1], "text": ["<sentence>", ...]}
"score" is always required: how well the Document answers the Query, as an integer from 0 (unrelated) to 100 (fully answers it).
BOTH evidence keys always follow it, in that order, whatever the Document is. A Document can be an image, written text, or a page that is both; give each key the evidence its own half of the Document supports and give the other one empty. Never omit a key and never add one.
"bbox_2d" is the ONE region of the image the answer is read off, as four integers on a 0-1000 grid, where 0,0 is the top-left corner and 1000,1000 the bottom-right. It must contain every piece of visual evidence the answer depends on, AND the surrounding context that makes that evidence readable: for a value in a table or chart, the row, column, axis or legend labels that say what it means; for an object, the caption or title that names it; for a comparison, a count or a spatial relation, every item involved. Prefer a slightly larger region over one that cuts context off, and do not give the whole image unless the answer genuinely depends on all of it. Give one box, never a list of boxes. If the Document has no image, or no region of it answers the Query, give "bbox_2d": [].
"text" is the sentences of the Document that answer the Query, copied verbatim -- do not shorten, merge, reorder or rewrite them, and do not add any of your own, and never describe the image here. Give EVERY sentence that is needed to answer the Query and no others: the Document is cut down to exactly this list, so a missing sentence is lost and a spare one prunes nothing. If the Document carries no written text, or no sentence of it answers the Query, give "text": [].

user: <Query>:{query}
<Document>:*(page image)*
{page text}

assistant: <think></think><|reranking|>{"score":

Figure 5: The full prompt of MMProvence. The shaded part is prefilled: an empty reasoning block, which disables thinking, the <|reranking|> token and the opening characters of the JSON object. The line breaks inside the reasoning block are omitted.

{"score": <integer>, "bbox_2d": [x0, y0, x1, y1], "text": [<sentence number>, ...]}

"text" is the NUMBERS of the sentences of the Document that answer the Query. Every sentence of the written text is preceded by its number in square brackets -- [1], [2], [3] and so on -- and "text" lists those numbers as integers in increasing order, e.g. "text": [2, 5]. Never copy, quote or describe a sentence here, and never give a number the Document does not show. Give the number of EVERY sentence that is needed to answer the Query and no others: the Document is cut down to exactly these sentences, so a missing number loses evidence and a spare one prunes nothing. If the Document carries no written text, or no sentence of it answers the Query, give "text": [].

Figure 6: The schema line (top) and the rule for `text` (bottom) of the sentence-index variant of MMProvence, which replace the corresponding lines of Figure 5; the rest of the prompt is unchanged.

Generator and judge. Both prompts consist of a single user turn (Figure 8). The generator receives its instruction, then each of the $k = 5$ pages as its image followed by its text, in rank order, and finally the question. A pruned page is passed in the same way with its crop and its pruned text, and a page whose text is pruned to nothing contributes its image only. The judge receives the question, up to five gold answers joined by semicolons, and the prediction, of which only the text after the generator's reasoning trace is kept; predictions and gold answers are truncated to 500 words and 1,000 tokens. The judge does not see the pages. Both models run with thinking enabled, and the judge's score is read after its reasoning trace.

system (extractive): You prune retrieved context for a retrieval-augmented generation system. Given a Query and a Document (its OCR/markdown text and/or its page image), identify only the text spans and the image region needed to answer the Query, and discard everything else. Copy kept text verbatim from the Document -- do not paraphrase or add anything not already present.

system (abstractive): You compress retrieved context for a retrieval-augmented generation system by summarizing, not by quoting. Given a Query and a Document (its OCR/markdown text and/or its page image), write a thorough summary, in your own words, of the information in the Document that could help answer the Query. Err on the side of including more detail rather than less -- if you are unsure whether a fact is relevant, include it rather than omit it. The summary may restate, combine, or rephrase facts from the Document, but must not introduce any fact that is not present in it.

user: <Instruct>: {instruction}
<Query>:{query}
<Document>:*(page image)*{page text}
Respond in exactly this format:
<kept_text> or <summary>
{output description}
</kept_text> or </summary>
<relevant_region>
...FULL if the whole page image is relevant, NONE if no part of the image is relevant, or x1,y1,x2,y2 (normalized 0-1 floats, the top-left/bottom-right corners of the tightest box enclosing only the relevant region) otherwise...
</relevant_region>

instruction (extractive): Keep only the text spans and image region relevant to answering the query.
output description (extractive): ...the exact relevant text copied verbatim from the Document, or NONE if no text is relevant...

instruction (abstractive): Summarize the information in the document that could help answer the query, in your own words. When uncertain whether something is relevant, include it rather than leave it out.
output description (abstractive): ...a thorough summary, in your own words, of the information in the Document that could help answer the Query -- when uncertain whether something is relevant, include it rather than leave it out -- or NONE if nothing in the Document is relevant...

Figure 7: The prompts of the prompted pruners. The extractive pruner answers in a <kept_text> tag and the abstractive pruner in a <summary> tag; everything else in the user turn is shared.

generator, user: You are a helpful assistant. Use the provided document pages (images and any text) to answer the question. If the answer is not present, say you don't know.
(page image 1) Document 1: {page text 1}
...
(page image 5) Document 5: {page text 5}
Question: {question}
Answer:

judge, user: You are grading a question-answering system. Given the question, the gold reference answer(s), and the system's predicted answer, judge whether the prediction is correct. Answer with a single digit: 1 if correct, 0 if incorrect.

Question: {question}
Gold answer(s): {gold answers}
Predicted answer: {prediction}
Score (0 or 1):

Figure 8: The prompts of the generator (top) and of the LLM judge (bottom).