



PIXELRAG: Web Screenshots Beat Text for Retrieval-Augmented Generation

Yichuan Wang^{*1} Zhifei Li^{*2,5} Zirui Wang¹ Paul Teilletche³ Lesheng Jin⁴
Matei Zaharia^{†1} Joseph E. Gonzalez^{†1} Sewon Min^{†1}

¹UC Berkeley ²Princeton University ³EPFL ⁴Databricks ⁵Renmin University of China
yichuan_wang@berkeley.edu zhifei.li@princeton.edu

Abstract

Augmenting large language models (LLMs) with retrieved web text has become a dominant paradigm, yet the web is not natively textual: existing systems depend on complex parsing pipelines that linearize HTML and discard layout, visual structure, and formatting. We introduce **PIXELRAG**, a new retrieval-augmented method that represents websites in their *native visual form* and performs retrieval and reading entirely in pixel space, enabling an end-to-end architecture that eliminates text abstraction. **PIXELRAG** is, to our knowledge, the first pipeline to operate over a full Wikipedia corpus in this form, scaling to a datastore of 30 million screenshot images with an efficient visual retrieval index. Built on an existing visual embedding model (i.e., Qwen3-VL-Embedding), **PIXELRAG** further fine-tunes this model on screenshot data with carefully curated contrastive training data. Retrieved screenshots are then fed directly as pixel inputs to a VLM, without intermediate text conversion. **PIXELRAG** consistently outperforms both no-retrieval and text-based RAG baselines, most surprisingly on widely studied text-centric tasks such as NQ and SimpleQA. It also achieves strong gains on multimodal open-domain QA (e.g., MMSearch), benchmarks over noisy news corpora (e.g., LiveVQA), and agentic benchmarks (e.g., MoNaCo), improving accuracy by up to 18.1% over text-based baselines. Finally, pixel representations enable a new efficiency lever for RAG through image compression, achieving up to 3× token cost reduction at lower resolutions while maintaining accuracy. Our results challenge the necessity of text representations in web retrieval, suggesting that web RAG can operate directly in the web’s native visual form while improving both performance and efficiency. Our code is available at <https://github.com/StarTrail-org/PixelRAG>.

1 Introduction

Retrieval-augmented generation (RAG) has become a dominant paradigm for grounding large language models (LLMs) in external knowledge, powering open-domain question answering, search-augmented agents, and deep-research systems [1–6]. Among retrieval sources, the web is the largest and most diverse knowledge base: an increasing number of LLM applications rely on web retrieval to stay current and factually grounded [7–9].

Text-based RAG assumes access to clean text documents for retrieval and reading, but hinges on a critical, often overlooked step: HTML parsing. Clean text is not free: HTML-to-text extraction

^{*}Equal contribution.

[†]Equal advising.

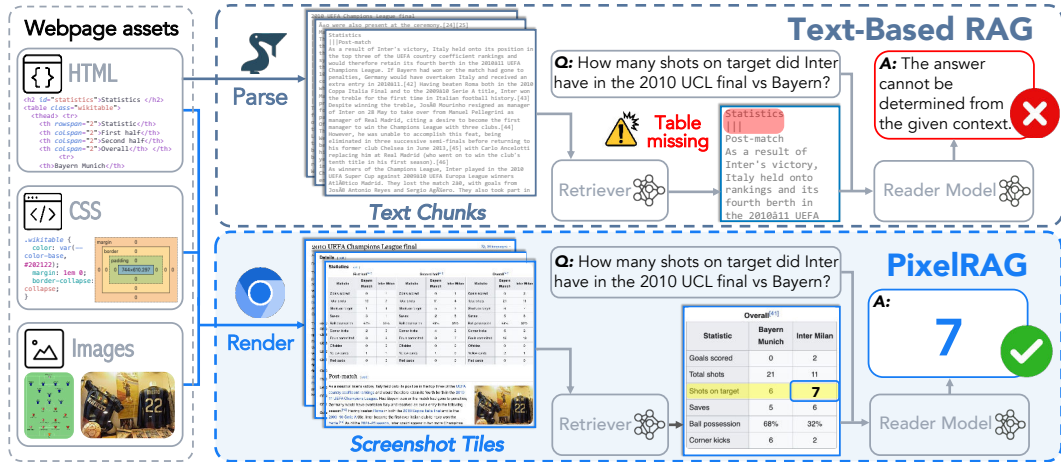


Figure 1: **Overview of PIXELRAG.** Text-based RAG (top) parses HTML into a text index and retrieves text chunks for the reader model. **PIXELRAG** (bottom, ours) renders each webpage, builds a visual index, and retrieves screenshot tiles for the reader — no parser required, fully visual.

is complex, heavily engineered, and error-prone [10–12]. Even state-of-the-art parsers [13–15] are brittle and inherently lossy, discarding visual cues (e.g., emphasis, layout) and structured content such as tables, charts, and images. These losses cascade: the retriever indexes a degraded view of the page, and the reader must reason over linearized, noisy text that may be difficult to comprehend.

In this work, we ask: can RAG operate directly in pixel space—on the web as users see it—bypassing complex and lossy HTML parsing? Concretely, this would index web screenshots, retrieve pages as images, and feed them to a VLM as context. While ambitious, recent advances in VLMs that significantly improved visual understanding, enabling capabilities such as OCR, document, and chart understanding, make this approach feasible [16–21]. As information-seeking increasingly involves structured and visual content, this approach also enables a unified representation without separate text and vision representations. Prior work has explored this paradigm in curated, small-scale domains such as PDFs [22–27], but it remains largely unexplored for the web—the dominant source of open-domain knowledge.

We introduce **PIXELRAG**, to our knowledge, the first end-to-end screenshot-based RAG approach at web scale, operating over the full Wikipedia corpus (7M articles) and news corpora from CNN, AP News, and BBC. While scalable data collection is non-trivial, we build a pipeline that first collects rendering assets (HTML, CSS, images) and reconstructs pages locally to produce screenshots at scale. Retrieval is powered by a Qwen3-VL-Embedding model [28], which we further adapt to screenshot data via efficient contrastive fine-tuning with synthetic QA pairs and dynamic hard-negative mining [29, 3]. Retrieved images are then fed directly as pixel inputs to a VLM reader, bypassing intermediate text representations.

Our experiments show that **PIXELRAG** consistently outperforms no retrieval and text-based RAG baselines across benchmarks, most surprisingly on widely used text-centric Wikipedia QA tasks such as NQ [30] and SimpleQA [31], where most, if not all, questions are answerable from text alone. **PIXELRAG** also delivers strong gains on multimodal QA, benchmarks over noisy news corpora (LiveVQA [32]), and agentic benchmarks [33]. Further analysis suggests that gains on text-centric tasks arise from leveraging 2D structural cues (e.g., tables, infoboxes) and avoiding information loss observed in HTML-to-text parsing. Notably, these gains scale with VLM capability, suggesting further headroom as models continue to improve.

Finally, pixel representations introduce a new, underexplored dimension for improving token efficiency in RAG via image compression, in line with recent work [34, 35]. Training the VLM reader to tolerate lower resolutions further improves the accuracy–efficiency Pareto frontier, enabling up to 3× token cost reduction while maintaining accuracy. To summarize, our contributions are fourfold:

1. We introduce PIXELRAG, the first end-to-end pixel-space RAG system at web scale, operating directly over rendered screenshots and bypassing HTML-to-text parsing. This simplifies the pipeline, preserves visual cues, and avoids information loss.
2. We develop a scalable pipeline for constructing screenshot-based corpora from the web, along with a visual embedding model trained on screenshot data via synthetic QA pairs and hard negatives.
3. We show that pixel-space retrieval on web data can consistently outperform text-based baselines across diverse benchmarks, including well-studied text-centric ones (e.g., NQ, SimpleQA).
4. We identify a new efficiency dimension via image compression, achieving up to $3\times$ token cost reduction. These gains suggest an emerging shift toward pixel-space RAG as multimodal models continue to improve.

2 Related Work & Motivation

Retrieval-Augmented Generation (RAG). RAG operates over large corpora of text documents, using a *retrieval model* to identify relevant text and a *reader model* to generate grounded answers [3, 1, 2], powering search-augmented agents [5, 36] and deep research systems [37]. In open-domain settings, these corpora are predominantly web-derived, including knowledge sources such as Wikipedia.

Despite this reliance on web data, most prior work assumes that clean textual documents are readily available, largely abstracting away the upstream HTML-to-text parsing step that converts raw web pages into the textual representations ultimately consumed by retrieval and reader models.

Text Parsing from Web Data. Text-based RAG over web data, therefore, relies on HTML-to-text parsers as a standard preprocessing layer for constructing retrievable text corpora. In large-scale data curation, reliable web parsing has long been recognized as a critical challenge [38, 11]: web pages vary enormously in structure, interleaving natural language passages with tables, infoboxes, figures, charts, and dynamically rendered content. Current pipelines typically rely on heuristic extractors such as *trafilatura* [13], *resiliparse* [39], and *mwparserfromhell* [40], or learned parsers such as ReaderLM [14] and Dripper [15].

However, even state-of-the-art parsers remain brittle and inherently lossy, often discarding visual cues (e.g., emphasis and layout) and structured content such as tables, charts, and images. Recent work shows that parser choice alone can substantially affect downstream performance [38, 11]: a single extractor may discard over 40% of recoverable webpage text [10]. Similar effects have also been observed in RAG systems, both in prior work [8] and in our own experiments (§5.2), where the gap between the two strongest parsing methods yields nearly a 10% absolute difference on SimpleQA.

Vision-Language Models for Text-Rich Images. At the same time, recent vision-language models (VLMs) are becoming increasingly capable of understanding text rendered as pixels, narrowing the gap with text-input models [41–43, 16]. Concurrently, systems such as DeepSeek-OCR [34] and Glyph [35] demonstrate that representing documents visually can sometimes be more token-efficient than fully textual pipelines. Together, these advances suggest that directly modeling rendered web pages as images may be a promising alternative to HTML-to-text parsing.

Visual Document Retrieval. Motivated by these advances, a growing line of work explores retrieving documents directly from their rendered visual representations [22, 44, 24], primarily on small-scale, visually rich benchmarks built from PDFs and slides [45–47]. Our work is inspired by this direction, but instead targets open-domain web retrieval—arguably the most dominant setting for modern RAG models. Compared to curated document collections, web corpora are vastly larger, more heterogeneous, and dominated by noisy, text-heavy pages rather than curated documents.

The closest prior work to ours are recent efforts to extend visual retrieval to Wikipedia [23, 25]; however, they remain limited to a small subset of Wikipedia and focus on retrieval rather than end-to-end RAG. In contrast, we study whether fully visual RAG can readily replace text-based RAG over large-scale Wikipedia, even on well-studied RAG benchmarks such as SimpleQA and NQ.

Summary: Why Pixel-Based RAG? To summarize, three observations motivate a shift from text-based to pixel-based RAG.

1. **Parsing discards critical information.** Even the best extractors strip images, charts, and rendered layout, and flatten or lose tables and other 2D structures—content that often contains the answer (Figure 1; Appendix D.2). Some content is only materialized during rendering, making it invisible to any HTML-level extractor.
2. **Text loses visual structure** Even when all content is preserved, converting a two-dimensional layout into a one-dimensional token sequence discards spatial grouping, font hierarchy, and emphasis. Web pages are designed to be consumed as rendered artifacts—these visual cues help readers locate information, and stripping them makes it harder for retrieval to distinguish answer-bearing regions from surrounding text. As shown in §5.2 and Appendix D.3, text retrieval often surfaces topically relevant but uninformative passages, whereas pixel-based retrieval localizes the answer directly.
3. **VLM advances increasingly favor pixel-based RAG.** Modern VLMs are becoming more capable and token-efficient, already outperforming text-only models on structurally rich content such as tables and infoboxes, making pixel-based retrieval increasingly practical and effective.

3 PIXELRAG

We present PIXELRAG, an end-to-end retrieval-and-generation pipeline over web data that operates entirely in pixel space (Figure 1). PIXELRAG consists of three stages: *data collection* (§3.1) renders webpages into screenshots and slices them into fixed-size tiles; *index construction* (§3.2) encodes each tile into a visual embedding and builds an approximate nearest-neighbor index; and *runtime retrieval and generation* (§3.3) retrieves the top- K tiles for a query and feeds them to a vision-language reader that produces the answer directly from pixels. We instantiate PIXELRAG on full Wikipedia as our running example. Wikipedia is a canonical corpus of web knowledge and the basis of most widely used text-based RAG pipelines and benchmarks, so building the same datastore in pixel space enables direct comparison against text-based RAG over the same underlying corpus. To our knowledge, PIXELRAG is the first system to operate over the full 7M-article Wikipedia corpus in pixel space. The pipeline is datastore-agnostic: §5.2 applies the same stages to a 668K-article news corpus.

3.1 Data Collection

Challenges. In text-based retrieval, building a datastore is well-studied: download HTML files and parse them into plain text. For pixel-based RAG, how to build a datastore is less clear. A naive approach is to open each webpage in a browser instance and capture a screenshot; however, this does not scale to millions of pages (e.g., roughly 30 days for all 7M articles). This is slow for two reasons: (a) online fetch of all webpage assets dominates wall-clock time (large network I/O), and (b) browsers are not designed for high-throughput rendering, limiting concurrency and causing frequent retries and stalls. Instead, we decouple fetching from rendering: we first fetch all sources into a local mirror, then render and tile entirely offline. This makes the pipeline scalable, reproducible, and fault-tolerant.

Fetching. Each corpus is first materialized into a local mirror. For Wikipedia, all assets (HTML, CSS, and images) are extracted from a pre-built Kiwix ZIM archive; for the news corpus, an asynchronous crawler fetches them with parallel crawls over different domains. Once cached, rendering proceeds entirely offline, making retries free and experiments reproducible. Per-source details are in Appendix A.2.

Rendering. Each cached page is rendered with Playwright (headless Chromium). We strip non-content elements (navigation bars, sidebars, surrounding whitespace; see Appendix A.1) to produce clean, content-only screenshots. Pages exceeding the browser’s single-shot viewport are captured by scrolling, so coverage is independent of page length.

Tiling. We fix the viewport width to 875 pixels (Wikipedia’s default content width) and slice each full-page screenshot into non-overlapping 1024-pixel-tall tiles (the last tile may be shorter). This yields $\sim 30\text{M}$ tiles for Wikipedia (7M articles) and $\sim 3.6\text{M}$ for the news corpus (668K articles).

3.2 Index Construction

Embedding. Related work in PDF retrieval (§2) largely relies on late-interaction multivector models such as ColPali, ColQwen [22], and Nemotron ColEmbed [44], which were found to be critical for capturing fine-grained information in document images. In our setting, however, there is a key difference: *scale*. Our datastore contains 30M tiles, far beyond the scale of prior visual-retrieval work (e.g., a few thousand pages [22, 47]), making multivector retrieval prohibitively expensive: each 875×1024 -pixel tile emits ~ 875 visual tokens¹ at 128 dimensions each (~ 112 K dimensions per tile), which would inflate the 30M-tile index to ~ 6.5 TB in fp16, far beyond single-host RAM. We therefore adopt single-vector retrieval: a single 2048-dimensional vector per tile keeps the full index at ~ 120 GB in fp16, manageable on one machine. Our results (§5) show that single-vector retrieval with a state-of-the-art visual embedding model is already sufficiently competitive. Here we use an off-the-shelf Qwen3-VL-Embedding-2B [28] and its fine-tuned variant (§4).

Index. At tens of millions of vectors, exact search is infeasible; we use a FAISS IVF index [48] for approximate nearest-neighbor search, which is fast to build and supports efficient addition and deletion of vectors, enabling incremental updates (e.g., ingesting new news articles daily or refreshing Wikipedia snapshots) without full re-indexing.

The full offline pipeline completes the 7M-article Wikipedia in ~ 2 days on a single machine (128 CPU cores, 2 TB RAM, 8 H100 GPUs).

3.3 Runtime Retrieval and Generation

At query time, the query is embedded with the same embedding model, the top- K tiles are retrieved by inner-product similarity, and the vision-language reader produces the answer directly from the retrieved pixels and the query (prompt templates in Appendix F). The quality–cost trade-off is governed by: (1) *Number of retrieved tiles K* : increasing K supplies the reader with more evidence but costs more visual tokens per query. (2) *Rendering resolution*: each tile is originally 875×1024 pixels, but can be downscaled (e.g., to 437×512) before being fed to the reader; since common open-source VLMs support dynamic resolution, visual-token count scales proportionally with pixel count, yielding a $\sim 4\times$ token reduction [16]. We study both trade-offs in §5.

4 Embedding Model Contrastive Learning Pipeline

Given a screenshot datastore, we want to fine-tune a visual embedding model tailored to it, improving retrieval accuracy over the base model. We describe a synthetic contrastive data generation recipe in §4.1 and the training procedure in §4.2.

4.1 Synthetic Contrastive Data Generation and Dynamic Hard-Negative Mining

Our goal is to curate a contrastive training set of $(q, p, \{n^-\})$ triples, where q is a query grounded in the visual content of a page p , and each n^- is a hard negative that shares page structure and topic with p but does *not* answer q . We use no external labeled data: the entire training set is synthesized from the datastore, with an LLM as the sole annotator. Note that the downstream benchmarks in §5 are *out-of-distribution*: the embedding model sees neither their queries nor their labels during training.

Stage 1: Synthetic query generation. We first filter out information-sparse pages such as listings and disambiguation pages, keeping only tiles that pass lightweight information-density heuristics (full method in Appendix A.3.1). Given a sampled tile p , we prompt an LLM to generate a natural-language query q whose answer is present in p (prompt in Appendix A.3.2). We then apply a two-part filter: (1) a *self-containedness* check discards queries that implicitly reference the source page (e.g., “...listed in the screenshot?”); (2) an *answerability* check re-prompts the LLM to answer q from p alone and keeps the pair only if correct (prompts in Appendix A.3.3, A.3.5).

Stage 2: Dynamic hard-negative mining with false-negative filtering. Given a (q, p) pair from Stage 1, we retrieve the top- K tiles for q using the base embedding model [29] and treat all except p

¹ $875 \times 1024 / 16^2 / 4 \approx 875$; 16 is the ViT patch size, and the VLM projector merges every 4 spatially aligned patches [16].

Query: "In what year was the '1972 Undefeated Team' inducted into the Miami Dolphins Ring of Honor?"

Proxy Embedding Model (base) Index → retrieve top-K

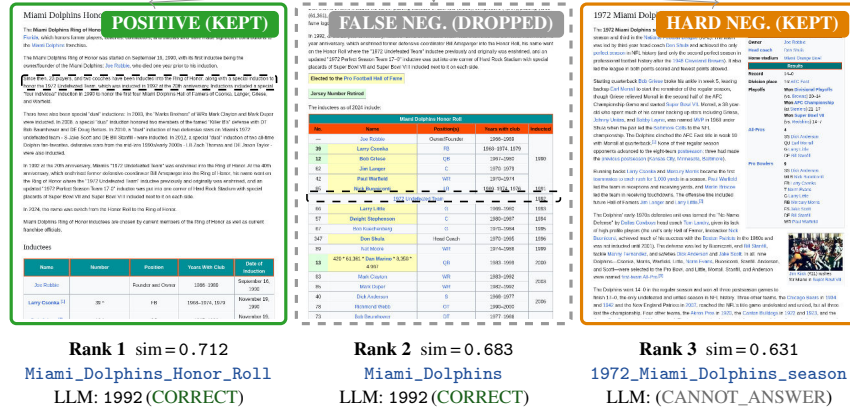


Figure 2: **Hard-negative mining with false-negative filtering.** The LLM correctly answers the query from both the positive tile (left, **kept**) and a second tile (center, **dropped as false negative**), but cannot answer from a topically adjacent page (right, **kept as hard negative**).

as hard-negative candidates. However, the same knowledge often appears in more than one page; a top- K neighbor may therefore also answer q , making it a false negative that penalizes a correct retrieval under the InfoNCE objective.

We address this with an LLM-based false-negative filter. For each top- K candidate (excluding p), we prompt the LLM to answer q from the candidate tile alone and judge the response as CORRECT, WRONG, or CANNOT_ANSWER (prompts in Appendix A.3.4). Candidates judged CORRECT are false negatives and dropped; the rest are kept as hard negatives. We retain the first M surviving candidates per query ($M=2$); Figure 2 walks through a worked example. This pipeline uses a strong LLM to distill clean supervision into the embedding model, improving retrieval without human labels.

4.2 Training Recipe

Contrastive training loss. We fine-tune the visual embedding model with InfoNCE loss [49] over mined triples and in-batch negatives. For a mini-batch of B triples $\{(q_i, p_i, \{n_{i,j}\}_{j=1}^M)\}_{i=1}^B$,

$$\mathcal{L} = -\frac{1}{B} \sum_{i=1}^B \log \frac{e^{\text{sim}(q_i, p_i) / \tau}}{e^{\text{sim}(q_i, p_i) / \tau} + \sum_{j=1}^M e^{\text{sim}(q_i, n_{i,j}) / \tau} + \sum_{k \neq i} e^{\text{sim}(q_i, p_k) / \tau}}, \quad (1)$$

with $\text{sim}(\cdot, \cdot)$ the cosine similarity and τ a temperature (full training configuration in Appendix A.3.6).

LoRA fine-tuning with unfrozen ViT. Prior visual retriever training, such as ColPali [22] and Swift [50], applies LoRA only to the LLM backbone, as adding LoRA to the ViT degrades performance on document images. We observe the opposite: we apply LoRA to both the LLM backbone and the ViT, and find it brings consistent gains on rendered webpage screenshots (Table 4). We hypothesize that visually near-duplicate webpage tiles require stronger visual discrimination, especially in fine-grained rendered text details and table structure, which benefits from ViT adaptation.

The full training completes in under 3 hours on a single H100, adding modest cost.

5 Evaluation

PIXELRAG outperforms all text-based RAG baselines across six benchmarks (§5.2). We ablate key design choices (§5.3), extend to agentic search (§5.4), study image compression as a cost knob (§5.5), and show that performance scales with VLM capability (§5.6).

Table 1: Recall@3 and end-to-end QA accuracy. The vertical line separates benchmarks using the Wikipedia corpus (left) and news corpus (right). *mwparserfromhell* is Wikipedia-specific; MMSearch lacks gold articles, so recall is marked –.

Method	NQ		NQ-Tables		SimpleQA		MMSearch		EVQA		LiveVQA	
	Recall	Acc	Recall	Acc	Recall	Acc	Recall	Acc	Recall	Acc	Recall	Acc
No retrieval	30.4		24.5		7.0		12.7		27.2		63.6	
<i>Text-based retrieval</i>												
<i>mwparserfromhell</i>	48.6	54.2	34.8	35.9	74.2	60.7	–	25.3	6.5	31.5		
Trafilatura	45.8	55.9	37.2	42.5	77.4	71.6	–	24.7	6.4	29.6	16.2	59.0
<i>Pixel-based retrieval</i>												
PIXELRAG (base)	53.5	57.9	45.5	47.0	80.8	73.8	–	28.3	27.1	40.7	38.9	70.3
PIXELRAG	58.8	58.7	51.1	48.8	83.8	78.8	–	28.3	35.0	45.1	33.3	70.0

5.1 Experimental Setup

Benchmarks. We evaluate on three task families across two corpora. (1) *Text-centric Wikipedia QA* includes NQ [30], NQ-Tables [51], and SimpleQA [31], which are widely studied benchmarks whose questions are primarily answerable from textual content. (2) *Multimodal Wikipedia QA* includes MMSearch [33]² and Encyclopedic VQA [52], consisting of image-grounded queries. (3) *News VQA* includes LiveVQA [32], which tests generalization beyond Wikipedia to a news corpus that is significantly noisier and more heterogeneous. The first five benchmarks query the Wikipedia datastore (30M tiles, §3.1); LiveVQA queries a separate news datastore (3.6M tiles). Dataset sizes and evaluation metrics are detailed in Appendix B.1.

Baselines. We compare against no-retrieval and two text-based RAG baselines that differ only in the HTML-to-text parser: *Trafilatura* [13], a general-purpose extractor ranked best in both prior work [11, 15] and our own comparison against six alternatives (Appendix D.1); and *mwparserfromhell* [40], a widely used Wikipedia-specific parser [53, 54]. Text indexes use 1024-token chunks, matching the ~ 875 visual tokens per tile (§3.2), with the same embedding model [28] and FAISS [48] index.

Configuration. Unless otherwise specified, the reader is Qwen3.5-4B [55] and receives $k=3$ retrieved items (text chunks or screenshot tiles). PIXELRAG (base) retrieves with the pretrained Qwen3-VL-Embedding-2B [28]; PIXELRAG adds contrastive fine-tuning from §4. Recall@ k measures whether any of the top- k retrieved items comes from the gold article (matched by URL).

5.2 End-to-End QA Results

PIXELRAG improves end-to-end QA accuracy across all six benchmarks (Table 1), most notably on text-centric tasks where questions are answerable from text alone. Even PIXELRAG (base) outperforms both text-based baselines on every task, with gains of up to 8.3% in recall and 4.5% in accuracy. Fine-tuning yields an additional 5.3% recall improvement and 5.0% accuracy gain. The largest improvements appear on NQ-Tables, where answering questions requires structured content such as tables and infoboxes. The gap widens further on multimodal VQA benchmarks. On EVQA, text retrieval provides little benefit over no retrieval, suggesting that retrieved text chunks are often off-target. In contrast, PIXELRAG improves QA accuracy by up to 18% while increasing retrieval recall on EVQA by over $5\times$. These gains suggest that preserving images and layout allows the embedding model to directly align visual queries with article content. This advantage extends beyond Wikipedia: on LiveVQA over our news corpus, PIXELRAG (base) outperforms the text baseline by 11.3% and more than doubles recall without domain-specific training. Wikipedia-trained fine-tuning does not transfer well to news domains; scaling training across domains remains future work.

Varying k and reader model. We vary $k \in \{1, 2, 3\}$ and the reader model on SimpleQA (Figure 3). PIXELRAG consistently outperforms text retrieval across all configurations: even PIXELRAG (base) beats both text baselines, and fine-tuning further widens the gap. At comparable token budgets, two PIXELRAG tiles match or exceed three text chunks while consuming fewer tokens, as a single tile

²MMSearch queries span diverse web topics; we use our Wikipedia datastore as the retrieval backend for all queries.

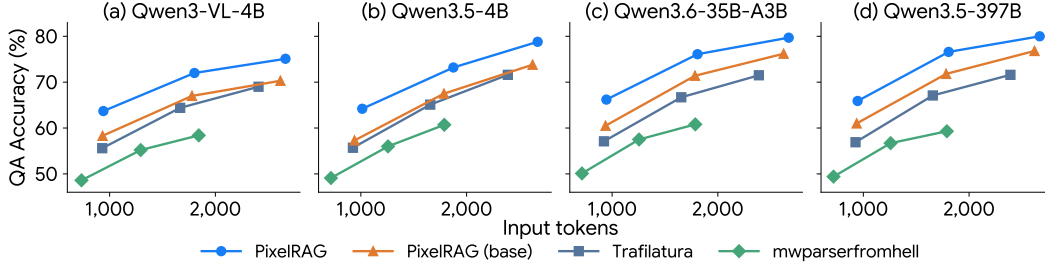


Figure 3: SimpleQA accuracy versus average input tokens across four reader models ($k \in \{1, 2, 3\}$).

Table 2: SimpleQA accuracy and Evidence Recall@3 by evidence type. Abbreviations: T. = Trafilatura, M. = mwparserfromhell.

Type	n	Accuracy (%)			Evidence Recall@3 (%)		
		PIXELRAG	T.	M.	PIXELRAG	T.	M.
Table	282	75.8 (+9.1)	66.7	48.2	34.8 (+11.0)	23.8	5.0
List	290	77.4 (+0.5)	76.9	62.8	36.9 (+8.3)	28.6	23.8
Infobox	503	83.3 (+4.6)	78.7	65.2	63.0 (-2.2)	65.2	54.9
Paragraph	571	79.4 (+7.9)	71.5	64.1	63.9 (+19.8)	44.1	38.7
Overall	1000 ³	78.8 (+7.1)	71.6	60.7	83.8 (+6.4)	77.4	74.2

packs denser evidence than a 1024-token chunk [34]. The advantage grows with stronger readers: at $k=2$, PIXELRAG leads Trafilatura by 9.4% with Qwen3.6-35B-A3B vs. 8.1% with Qwen3.5-4B.

Qualitative analysis: SimpleQA accuracy breakdown by answer location. We examine what each method actually retrieves using *evidence Recall@3*: whether the answer-bearing tile or chunk appears in the reader’s top-3, broken down by where the answer lives on the page (Table 2; definition in Appendix D.4). Overall, PIXELRAG retrieves answer-bearing evidence 6% more often than Trafilatura (83.8% vs. 77.4%), explaining most of the end-to-end accuracy gap.

The advantage concentrates on two evidence types. For *tables*, parsers frequently lose structured content during linearization, dropping evidence Recall@3 to 23.8% (Trafilatura) and 5.0% (mwparserfromhell) versus 34.8% for PIXELRAG. For *paragraphs*, the gap is even larger (+19.8%): once linearized, keyword-dense infobox text overlaps with nearly any factual query about the article, displacing the answer-bearing paragraph from top-3; the visual embedding model is immune because infoboxes have a visually distinct bordered-sidebar layout. Switching to mwparserfromhell widens the overall accuracy gap by a further 10.9%, confirming that the advantage is not parser-specific.

Text-based retrieval exhibits three distinct failure modes (Appendix D.4). *Parser loss* (36.6%): linearization destroys 2D structures such as tables, dropping the answer from the text corpus entirely; e.g., a match-statistics table is flattened to empty delimiter characters, leaving no chunk that preserves its cell values (Fig. 10). *Rank loss* (55.2%): answer-bearing text exists in the corpus but ranks outside the reader’s top-3. A common pattern in text retrieval is infobox displacement: the linearized infobox produces keyword-dense but answer-irrelevant text that matches nearly any factual query about the entity, occupying rank 1 while the answer paragraph falls to rank 60+. PIXELRAG is less susceptible because its visual encoder distinguishes the infobox’s bordered-sidebar layout from body text (Figs. 11–12). *Reader loss* (8.2%): even when evidence reaches top-3, linearizing tables and lists collapses the row and hierarchy grouping the reader needs; a multi-year honors list, for example, loses its year headings once flattened, causing the reader to attribute an entry to the wrong year (Fig. 13).

5.3 Ablation Studies

Retrieval and reader input modality ablation. To disentangle the contributions of retrieval modality and reader input format, we fix the embedding model to Qwen3-VL-Embedding-2B, then vary only these two factors across five configurations (Table 3): (1) **Screenshot** → **Screenshot** (PIXELRAG (base)); (2) **Screenshot** → **OCR**; (3) **Text** → **Rendered image**; (4) **Text** → **Text**

Table 3: Retrieval–reader modality ablation.

Retrieval	Reader	SimpleQA	LiveVQA
Screenshot	Screenshot	73.8	70.3
Screenshot	OCR text	72.2	64.5
Text	Rendered image	67.4	56.6
Text	Text	71.6	59.0
Text	HTML	59.8	56.6

Table 4: Embedding training recipe ablation. We use dynamic hard negatives with ViT unfreezing.

Recipe	ViT	QA
Base model	–	0.725
In-batch negatives	Frozen	0.710
Naive hard negatives	Frozen	0.723
Dynamic hard negatives	Frozen	0.770
Dynamic hard negatives (ours)	Unfrozen	0.793

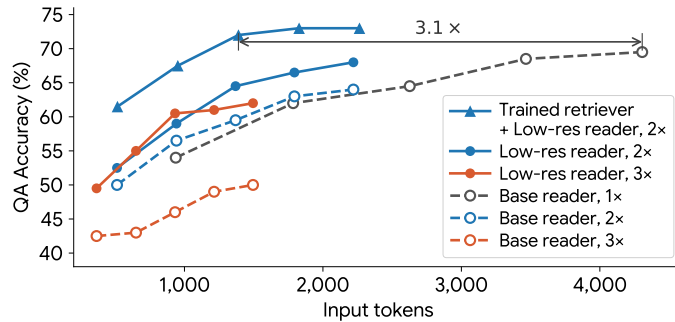
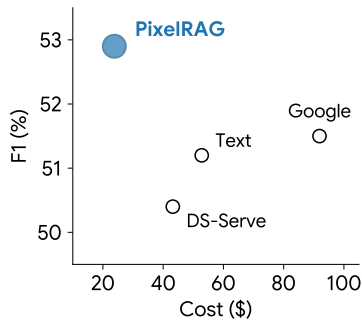


Figure 4: Agentic multi-hop QA on MoNaCo: F1 vs. total cost. Figure 5: SimpleQA accuracy vs. input tokens under image compression ($k \in \{1, \dots, 5\}$). $c \times$ denotes the ratio.

(standard text-based RAG); and (5) **Text** \rightarrow **HTML**. Both retrieval and reader input format contribute, but retrieval is the larger factor (Table 3). Screenshot retrieval is consistently better: the first two rows always outperform the text-retrieval variants, and Screenshot \rightarrow OCR outperforms Text \rightarrow Text on both benchmarks, showing that visual retrieval improves downstream accuracy even when the reader sees only text. We also test feeding raw HTML to the reader to preserve table and list structure (Text \rightarrow HTML), but this *underperforms* flat text on both benchmarks: HTML markup inflates context by $3.8 \times$, consuming the reader’s token budget on tags rather than evidence (Appendix D.5). Directly indexing and reading raw HTML fares no better, with accuracy dropping by up to 29% (Appendix D.6).

Retriever training recipe ablation. We ablate the visual embedding training recipe (§4) by varying the negative curation strategy and the ViT training mode (Table 4; setup in Appendix A.4). In-batch negatives alone slightly underperform the base model. Naive hard negatives (top-ranked non-positive passages from the base embedding model) improve over in-batch negatives but introduce noisy supervision from false negatives; dynamic hard-negative mining (§4), which filters these, accounts for the largest single jump (+4.7%). Finally, unlike prior work on document retrieval [22], unfreezing the ViT through LoRA further improves results: rendered web pages are more visually homogeneous than PDFs or slides, requiring finer visual discrimination than a frozen backbone provides.

5.4 Agentic Search with PIXELRAG

We evaluate PIXELRAG as the search backend for a GPT-5 ReAct agent on MoNaCo [56], a multi-hop Wikipedia QA benchmark (setup in Appendix B.1). PIXELRAG achieves the highest F1 at the lowest cost (Figure 4), outperforming text-chunk retrieval, Google [57], and DS-Serve [58] while costing 2–4 \times less. We report token-level F1, the official MoNaCo metric, as answers are often multi-value lists where partial credit is appropriate. Because screenshot tiles pack more information per retrieval step than text chunks, the agent requires fewer searches and accumulates less conversation history: 3.6M prompt tokens versus 37.5M for text retrieval (10 \times reduction).

5.5 Image Compression to Save Visual Tokens

In RAG systems, prefilling retrieved context dominates inference cost, and reducing context length directly lowers latency and expense. Pixel inputs expose a compression knob unavailable to text: image resolution. Downsampling tiles via Lanczos resampling [59] reduces token count proportionally

through VLM dynamic resolution [16], without changing the datastore or retriever: a $c\times$ compression scales each side by $1/\sqrt{c}$. To maintain accuracy at lower resolution, we fine-tune the reader (Qwen3-VL-4B) on compressed tiles. Training data is constructed at zero additional annotation cost by reusing the (q, p, a) triples from the embedding pipeline (§4.1), augmented with retrieved distractor tiles to simulate the inference setting (training details in Appendix A.5). At $2\times$ compression, the fine-tuned reader almost matches native-resolution accuracy while halving tokens (Figure 5). Combining the trained retriever with the $2\times$ -compressed reader reaches 72% at $k=3$, surpassing the uncompressed $k=5$ ceiling at one third of the token cost.

5.6 PIXELRAG Performance Scales with VLM Capability

Progress in VLM reading ability is a key driver of PIXELRAG’s performance (Table 5). Early VLMs struggle with screenshots: small models in the Llama-3.2-Vision and Qwen2-VL families trail text retrieval by over 12.5%. From Qwen3-VL-8B (▲) onward, PIXELRAG matches or beats text, and the advantage reaches up to 5.9%, consistent with VLMs becoming increasingly capable at understanding text in pixels [43]. Table 5 pinpoints the crossover where pixel retrieval first matches text at Qwen3-VL-4B (★, 70.5% vs. 69.0%): below this capability threshold models cannot reliably read text rendered in pixels and text retrieval dominates, while above it every model we test favors pixel retrieval, the gap widening to +4.6 pp for Qwen3.6-35B-A3B (◆). Concurrently, successive VLM generations encode the same tile into fewer input tokens, so the cost of pixel retrieval decreases with each model generation without any pipeline change (Table 5). These trends suggest that PIXELRAG’s pixel-native approach will continue to benefit from VLM progress, with the reader as the only component that needs upgrading—the retrieval pipeline and datastore remain unchanged.

Table 5: Reader-model sweep on SimpleQA. Shading: accuracy, input tokens.

Reader model	Pixel retrieval			Text retrieval		
	Acc	in tok	out tok	Acc	in tok	out tok
LLaVA-1.5-7B [60]	0.092	1817	32	0.504	2537	33
Llama-3.2-11B-Vision [61]†	0.534	6485	30	0.700	2252	26
Llama-3.2-90B-Vision [61]†	0.574	6485	32	0.670	2252	39
Llama-4-Scout-17B-16E [62]	0.618	5613	93	0.667	2175	60
Llama-4-Maverick-17B-128E [62]	0.756	5613	57	0.719	2175	73
Qwen2-VL-2B [63]	0.439	3446	27	0.564	2410	41
Qwen2-VL-7B [63]	0.628	3446	15	0.663	2410	26
Qwen2-VL-72B [63]	0.745	3446	23	0.718	2410	14
Qwen2.5-VL-3B [64]	0.460	3446	20	0.661	2410	17
Qwen2.5-VL-7B [64]	0.713	3446	16	0.678	2410	21
Qwen2.5-VL-32B [64]	0.717	3446	29	0.691	2410	56
Qwen2.5-VL-72B [64]	0.745	3446	17	0.694	2346	26
Qwen3-VL-2B [16]	0.613	2615	229	0.676	2408	34
★ Qwen3-VL-4B [16]	0.705	2615	70	0.690	2408	62
▲ Qwen3-VL-8B [16]	0.726	2615	95	0.697	2408	100
Qwen3-VL-30B-A3B [16]	0.733	2615	109	0.715	2344	37
Qwen3-VL-32B [16]	0.742	2615	52	0.710	2344	43
Qwen3-VL-235B-A22B [16]	0.759	2615	52	0.720	2408	42
Qwen3.5-0.8B [55]	0.602	2624	73	0.593	2390	50
Qwen3.5-2B [55]	0.667	2624	116	0.667	2390	104
Qwen3.5-4B [55]	0.738	2624	118	0.716	2459	95
Qwen3.5-9B [55]	0.750	2624	115	0.701	2390	83
Qwen3.5-27B [55]	0.759	2624	88	0.711	2390	69
Qwen3.5-35B-A3B [55]	0.737	2622	148	0.713	2390	89
Qwen3.6-27B [65]	0.754	2624	73	0.715	2390	65
◆ Qwen3.6-35B-A3B [65]	0.751	2624	110	0.705	2390	50
Qwen3.5-4B (reasoning) [55]	0.745	2622	1311	0.697	2457	1597
Qwen3.5-397B-A17B (reasoning) [55]	0.769	2622	1081	0.710	2461	1188
Qwen3.6-35B-A3B (reasoning) [65]	0.763	2622	1254	0.717	2457	1005

† Llama-3.2-Vision accepts only one image; pixel column uses $k=1$.

6 Conclusion

RAG over web data is critical for grounding LLMs, yet extracting clean text from diverse webpages remains a persistent challenge. PIXELRAG sidesteps this problem by performing retrieval and generation directly over rendered screenshots, with no HTML parsing or text extraction in the loop. We scale this approach to over 30M tiles covering all of Wikipedia, and show that this pixel-native pipeline outperforms text-based RAG even on text-centric benchmarks where no visual reasoning is needed. The advantage grows with VLM progress: newer models achieve higher accuracy with fewer tokens, and image compression provides up to $3\times$ token reduction, making PIXELRAG increasingly practical with each generation.

Limitations, broader impact, and future directions are discussed in Appendix E.

Acknowledgement

We thank Akari Asai for sharing knowledge about the MoNaCo dataset, Zhiying Xu for support during the early stages of the project, Lisa Dunlap for inspiring discussions on benchmark selection, Rulin Shao for insightful discussions throughout the project and support during the release phase, and Xueguang Ma for providing a naive version of screenshot capture code that helped bootstrap the project. We also thank members of Sky Lab and the SM group for helpful discussions and proofreading. This work was supported in part by the Gemini Academic Program and the NVIDIA Academic Grant Program. This research was also supported by gifts from Accenture, Amazon, AMD, Anyscale, Broadcom Inc., Google, IBM, Intel, Intesa Sanpaolo, Lambda, Mibura Inc, Samsung SDS, and SAP.

References

- [1] 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.
- [2] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. REALM: Retrieval-augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938, 2020.
- [3] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6769–6781, 2020.
- [4] Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 874–880, 2021.
- [5] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.
- [6] Xinxin Lyu, Michael Duan, Rulin Shao, Pang Wei Koh, and Sewon Min. Frustratingly simple retrieval improves challenging, reasoning-intensive benchmarks. *arXiv preprint arXiv:2507.01297*, 2025.
- [7] Rulin Shao, Jacqueline He, Akari Asai, Weijia Shi, Tim Dettmers, Sewon Min, Luke Zettlemoyer, and Pang W Koh. Scaling retrieval-based language models with a trillion-token datastore. *Advances in Neural Information Processing Systems*, 37:91260–91299, 2024.
- [8] Alex Fang, Thomas Voice, Ruoming Pang, Ludwig Schmidt, and Tom Gunter. Reusing pre-training data at test time is a compute multiplier. *arXiv preprint arXiv:2511.04234*, 2025.

- [9] Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, et al. Websailor: Navigating super-human reasoning for web agent. *arXiv preprint arXiv:2507.02592*, 2025.
- [10] Jeffrey Li, Joshua P Gardner, Doug Kang, Fangping Shi, Karanjeet Singh, Chun-Liang Li, Herumb Shandilya, David Leo Wright Hall, Oncel Tuzel, Percy Liang, et al. Beyond a single extractor: Re-thinking html-to-text extraction for llm pre-training. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 5836–5861, 2026.
- [11] Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.
- [12] Jiejun Tan, Zhicheng Dou, Wen Wang, Mang Wang, Weipeng Chen, and Ji-Rong Wen. HTML-RAG: HTML is better than plain text for modeling retrieved knowledge in RAG systems. In *Proceedings of the ACM Web Conference 2025 (WWW)*, 2025.
- [13] Adrien Barbaresi. Trafilatura: A web scraping library and command-line tool for text discovery and extraction. In Heng Ji, Jong C. Park, and Rui Xia, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 122–131, Online, August 2021. Association for Computational Linguistics.
- [14] Feng Wang, Zesheng Shi, Bo Wang, Nan Wang, and Han Xiao. ReaderLM-v2: Small language model for HTML to markdown and JSON. *arXiv preprint arXiv:2503.01151*, 2025.
- [15] Mengjie Liu, Jiahui Peng, Wenchang Ning, et al. Dropper: Token-efficient main HTML extraction with a lightweight LM. *arXiv preprint arXiv:2511.23119*, 2025.
- [16] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.
- [17] OpenAI. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.
- [18] Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.
- [19] Anthropic. The Claude 3 model family: Opus, Sonnet, Haiku. Technical report, 2024. Technical Report.
- [20] Jinguo Zhu, Weiyun Wang, Zhe Chen, et al. InternVL3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.
- [21] Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, et al. Charxiv: Charting gaps in realistic chart understanding in multimodal llms. *Advances in Neural Information Processing Systems*, 37:113569–113697, 2024.
- [22] 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*, 2025.
- [23] Xueguang Ma, Sheng-Chieh Lin, Minghan Li, Wenhui Chen, and Jimmy Lin. Unifying multimodal retrieval via document screenshot embedding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6492–6505, 2024.
- [24] Shi Yu, Chaoyue Tang, Bokai Xu, Junbo Cui, Junhao Ran, Yukun Yan, Zhenghao Liu, Shuo Wang, Xu Han, Zhiyuan Liu, and Maosong Sun. VisRAG: Vision-based retrieval-augmented generation on multi-modality documents. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

- [25] Jaemin Cho, Debanjan Mahata, Ozan Irsoy, Yujie He, and Mohit Bansal. M3docrag: Multimodal retrieval is what you need for multi-page multi-document understanding. *arXiv preprint arXiv:2411.04952*, 2024.
- [26] Yibo Yan, Jiahao Huo, Guanbo Feng, Mingdong Ou, Yi Cao, Xin Zou, Shuliang Liu, Yuanhuiyi Lyu, Yu Huang, Jungang Li, et al. Unlocking multimodal document intelligence: From current triumphs to future frontiers of visual document retrieval. *arXiv preprint arXiv:2602.19961*, 2026.
- [27] Jiahao Huo, Yu Huang, Yibo Yan, Ye Pan, Kening Zheng, Wei-Chieh Huang, Yi Cao, Mingdong Ou, Philip S. Yu, and Xuming Hu. CausalEmbed: Auto-regressive multi-vector generation in latent space for visual document embedding. *arXiv preprint arXiv:2601.21262*, 2026.
- [28] Mingxin Li, Yanzhao Zhang, Dingkun Long, Keqin Chen, Sibao 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.
- [29] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*, 2021.
- [30] 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.
- [31] Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. Measuring short-form factuality in large language models. *arXiv preprint arXiv:2411.04368*, 2024.
- [32] Mingyang Fu, Yuyang Peng, Dongping Chen, Zetong Zhou, Benlin Liu, Yao Wan, Zhou Zhao, Philip S. Yu, and Ranjay Krishna. Seeking and updating with live visual knowledge. *arXiv preprint arXiv:2504.05288*, 2025.
- [33] Dongzhi Jiang, Renrui Zhang, Ziyu Guo, Yanmin Wu, Jiayi Lei, Pengshuo Qiu, Pan Lu, Zehui Chen, Chaoyou Fu, Guanglu Song, Peng Gao, Yu Liu, Chunyuan Li, and Hongsheng Li. MMSearch: Benchmarking the potential of large models as multi-modal search engines. *arXiv preprint arXiv:2409.12959*, 2024.
- [34] Haoran Wei, Yaofeng Sun, and Yukun Li. Deepseek-ocr: Contexts optical compression. *arXiv preprint arXiv:2510.18234*, 2025.
- [35] Jiale Cheng, Yusen Liu, Xinyu Zhang, Yulin Fei, Wenyi Hong, Ruiliang Lyu, Weihang Wang, Zhe Su, Xiaotao Gu, Xiao Liu, et al. Glyph: Scaling context windows via visual-text compression. *arXiv preprint arXiv:2510.17800*, 2025.
- [36] Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.
- [37] Tongyi DeepResearch Team, Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Huifeng Yin, Jialong Wu, Jingren Zhou, et al. Tongyi deepresearch technical report. *arXiv preprint arXiv:2510.24701*, 2025.
- [38] Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. Datacomp-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282, 2024.
- [39] Janek Bevendorff, Benno Stein, Matthias Hagen, and Martin Potthast. Elastic ChatNoir: Search Engine for the ClueWeb and the Common Crawl. In Leif Azzopardi, Allan Hanbury, Gabriella Pasi, and Benjamin Piwowarski, editors, *Advances in Information Retrieval. 40th European Conference on IR Research (ECIR 2018)*, Lecture Notes in Computer Science, Berlin Heidelberg New York, March 2018. Springer.

- [40] Ben Kurtovic and contributors. mwparserfromhell: A python parser for mediawiki wikicode. <https://github.com/earwig/mwparserfromhell>, 2026.
- [41] Yujie Lu, Xiujun Li, Tsu-Jui Fu, Miguel Eckstein, and William Yang Wang. From text to pixel: Advancing long-context understanding in mllms. *arXiv preprint arXiv:2405.14213*, 2024.
- [42] Kaiser Sun, Xiaochuang Yuan, Hongjun Liu, Chen Zhao, Cheng Zhang, Mark Dredze, and Fan Bai. Reading, not thinking: Understanding and bridging the modality gap when text becomes pixels in multimodal llms. *arXiv preprint arXiv:2603.09095*, 2026.
- [43] Zhiheng Lyu, Xueguang Ma, and Wenhui Chen. Pixelworld: How far are we from perceiving everything as pixels? *arXiv preprint arXiv:2501.19339*, 2025.
- [44] Gabriel de Souza P Moreira, Ronay Ak, Mengyao Xu, Oliver Holworthy, Benedikt Schifferer, Zhiding Yu, Yauhen Babakhin, Radek Osmulski, Jiarui Cai, Ryan Chesler, et al. Nemotron colembed v2: Top-performing late interaction embedding models for visual document retrieval. *arXiv preprint arXiv:2602.03992*, 2026.
- [45] Quentin Macé, António Loison, and Manuel Faysse. Vidore benchmark v2: Raising the bar for visual retrieval. *arXiv preprint arXiv:2505.17166*, 2025.
- [46] Fengbin Zhu, Ziyang Liu, Xiang Yao Ng, Haohui Wu, Wenjie Wang, Fuli Feng, Chao Wang, Huanbo Luan, and Tat Seng Chua. Mmdocbench: Benchmarking large vision-language models for fine-grained visual document understanding. *arXiv preprint arXiv:2410.21311*, 2024.
- [47] Connor Shorten, Augustas Skaburskas, Daniel M Jones, Charles Pierse, Roberto Esposito, John Trengrove, Etienne Dilocker, and Bob van Luijt. Irapapers: A visual document benchmark for scientific retrieval and question answering. *arXiv preprint arXiv:2602.17687*, 2026.
- [48] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2021.
- [49] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.
- [50] Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Hong Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. Swift:a scalable lightweight infrastructure for fine-tuning, 2024.
- [51] Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Martin Eisenschlos. Open domain question answering over tables via dense retrieval. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 512–519, 2021.
- [52] Thomas Mensink, Jasper Uijlings, Lluís Castrejon, Arushi Goel, Felipe Cadar, Howard Zhou, Fei Sha, André Araujo, and Vittorio Ferrari. Encyclopedic VQA: Visual questions about detailed properties of fine-grained categories. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3082–3092, 2023.
- [53] Tristan Thrush, Helen Ngo, Nathan Lambert, and Douwe Kiela. Online language modelling data pipeline. <https://github.com/huggingface/olm-datasets>, 2022.
- [54] NeuML. Neuml/wikipedia: Wikipedia text dataset. <https://huggingface.co/datasets/NeuML/wikipedia>, 2024. Text extracted from Wikipedia XML dumps via mwparserfromhell.
- [55] Qwen Team. Qwen3.5: Towards native multimodal agents. <https://qwen.ai/blog?id=qwen3.5>, 2026.
- [56] Tomer Wolfson, Harsh Trivedi, Mor Geva, Yoav Goldberg, Dan Roth, Tushar Khot, Ashish Sabharwal, and Reut Tsarfaty. MoNaCo: More natural and complex questions for reasoning across dozens of documents. *Transactions of the Association for Computational Linguistics*, 2025.

- [57] SerpApi. SerpApi: Google search api. <https://serpapi.com>, 2025.
- [58] Jinjian Liu, Yichuan Wang, Xinxi Lyu, Rulin Shao, Joseph E. Gonzalez, Matei Zaharia, and Sewon Min. DS SERVE: A framework for efficient and scalable neural retrieval. In *Fortieth AAAI Conference on Artificial Intelligence (AAAI)*, pages 41631–41633, 2026.
- [59] Wikipedia contributors. Lanczos resampling. https://en.wikipedia.org/wiki/Lanczos_resampling, 2025. Accessed: 2026-04-29.
- [60] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2023.
- [61] Meta AI. Llama 3.2: Lightweight text and multimodal models. <https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/>, 2024.
- [62] Meta AI. The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation. <https://ai.meta.com/blog/llama-4-multimodal-intelligence/>, 2026. Blog post.
- [63] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-VL: Enhancing vision-language model’s perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.
- [64] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibong Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.
- [65] Qwen Team. Qwen3.6: Towards real world agents. <https://qwen.ai/blog?id=qwen3.6/>, 2026.
- [66] Yichuan Wang, Zhifei Li, Shu Liu, Yongji Wu, Ziming Mao, Yilong Zhao, Xiao Yan, Zhiying Xu, Yang Zhou, Ion Stoica, et al. Leann: A low-storage vector index. *arXiv preprint arXiv:2506.08276*, 2025.
- [67] Tobias Weyand, André Araujo, Bingyi Cao, and Jack Sim. Google Landmarks Dataset v2 — a large-scale benchmark for instance-level recognition and retrieval. In *CVPR*, 2020.
- [68] OpenAI. Introducing GPT-4.1 in the API, April 2025. Blog post.
- [69] Kiwix Association. Kiwix — offline reader for web content. <https://kiwix.org>, 2007. Open-source offline browser using the ZIM archive format.

Technical appendices and supplementary material

Contents

<i>A System & Implementation Details</i>	
A.1 Rendering Pipeline	16
A.2 Datastore Fetching	16
A.3 Embedding Training: Data Recipe Details and Prompts	18
A.4 Reader Fine-tuning: SFT Results across Compression Ratios	19
<i>B Evaluation Protocol</i>	
B.1 Benchmark Details	19
B.2 Grading Protocol	20
<i>C Additional Results</i>	
C.1 Results with Alternative Readers	21
C.2 Full Results of Scaling VLM Performance	21
<i>D Analysis</i>	
D.1 Wikipedia Text Extractor Comparison	21
D.2 Visual Information Loss During HTML Parsing	22
D.3 Retrieval Signal Loss Under Text Linearization	22
D.4 Detailed Failure Decomposition	23
D.5 HTML DOM Lookup Baseline: Setup and Analysis	26
D.6 Directly RAG on Raw HTML Data	30
<i>E Limitations, Broader Impact, and Future Work</i>	
	31
<i>F Prompt Listings</i>	
	32

A System & Implementation Details

A.1 Rendering Pipeline

We render each Wikipedia page using Playwright in a headless Chromium browser at a fixed viewport width. After capture, we strip browser UI elements (navigation bars, sidebars, toolbars) and crop away surrounding whitespace, retaining only the article content area. This produces a clean, content-only screenshot that is tiled and indexed into PIXELRAG. Figure 6 shows an example before and after this processing step.

Storage considerations. The rendered tile images can be sizable (~ 5.6 TB for Wikipedia and ~ 469 GB for news). In principle, the tile images need not be stored persistently: after embedding, one can retain only the vector index and re-render the top- K pages on the fly from their original HTML, CSS, and JavaScript sources at query time, eliminating the image storage cost entirely. The vector index itself can be further compressed with low-storage indexing techniques such as LEANN [66], reducing the embedding storage cost as well.

A.2 Datastore Fetching

Wikipedia. We populate the local cache from a Kiwix-serve instance backed by a ZIM archive (dump date: 2025-08). Kiwix exposes the full Wikipedia corpus as a static snapshot, which avoids network bottlenecks entirely during rendering and sidesteps potential legal concerns associated with large-scale crawling of live pages. After excluding redirects, we render 7,134,778 articles (100% coverage of content pages), yielding approximately 30M tiles at ~ 4.2 tiles per article.

News corpus. We collect articles from three major English-language news outlets: BBC (356,358 articles), AP News (261,241 articles), and CNN (49,924 articles), totaling 667,523 articles and 3.6M screenshot tiles.

A.3 Embedding Training: Data Recipe Details and Prompts

This appendix collects the full prompts, heuristics, and implementation details behind the embedding-training pipeline of §4. Each sub-subsection is referenced from the exact point of choice in the main text.

A.3.1 Knowledge-intensive tile sampling

Referenced from §4.1, Stage 1. Not every Wikipedia page, or every tile within a page, is suitable for synthetic query generation. We apply three successive filters to select a pool of knowledge-intensive tiles.

Render-quality gate. We drop any page whose offline render is too small or failed: `page_height < 3,000 px`. This removes empty pages, redirects, and partial renders before they are ever sampled.

Title/URL regex blacklist. We exclude meta and namespace pages by title and URL pattern matching: disambiguation pages, pages in the `Category:`, `Template:`, `Wikipedia:`, `Portal:`, `File:`, `Help:`, `Talk:`, `Module:`, and `Draft:` namespaces, as well as `*_deaths` and `*_births` list pages.

Chunk-position filter. Within a page that passes both filters, we sample only from the first 70% of tiles, so that footers, reference lists, and “See also” boilerplate never become positive evidence.

A.3.2 Synthetic query generation prompt

Referenced from §4.1, Stage 1. We prompt `gemini-3.1-flash-lite-preview` (Vertex AI) with the rendered tile and the prompt in Figure 15, decoding at `temperature=0.7` and `max_output_tokens=1024`. The model returns either the five-line `Q / A / S / T / C` block or the literal string `SKIP`.

Figure 14 shows a concrete example; the full prompt is in Figure 15 (Appendix F).

A.3.3 Self-contained-query filter prompt

Referenced from §4.1, Stage 1 (first false-positive filter). After generation, we ask `gpt-4o` (OpenAI) at `temperature=0.0` to label each query as self-contained (YES) or not (NO), batching 50 queries per request. On our canonical training set this filter drops 15.1% of candidate pairs (195,079 → 165,537). The prompt is given in Figure 16 (Appendix F).

A.3.4 Hard-negative consistency-judge prompt

Referenced from §4.1, Stage 2. For each (q, p) pair, we retrieve the top- K candidates of q from the base embedding model ($K=20$ in our pipeline), skip the positive, and walk the remaining candidates in rank order. For each candidate, we run a two-stage filter that takes the candidate tile as the only visual context. **Stage A** (*answer*, Figure 17) shows the candidate tile and the query to GPT-4o and asks for a short answer or the literal string `CANNOT_ANSWER`. **Stage B** (*judge*, Figure 17) shows the same tile, the query, and Stage A’s answer to GPT-4o, and classifies the candidate answer as exactly one of `CORRECT`, `WRONG`, or `CANNOT_ANSWER`. A `CORRECT` verdict means the candidate truly answers the query and is a false negative; we drop it. `WRONG` or `CANNOT_ANSWER` means the candidate is visually or topically related but does not actually answer the query, which is the hard-negative signal we want; we keep it. We retain the first M candidates that pass both stages; if the top- K pool yields fewer than M surviving hard negatives, the entire (q, p) example is dropped from the training set. Full prompts are in Figures 17–17 (Appendix F).

A.3.5 Answerability filter (Stage 1, second false-positive filter)

Referenced from §4.1, Stage 1 (second false-positive filter). The answerability filter reuses the same two prompts as the consistency judge above (Figures 17 and 17); the only difference is that the visual context is the *positive* tile p rather than a candidate tile. Concretely, we run Stage A on (q, p) to obtain a candidate answer a^+ . If a^+ equals the literal string `CANNOT_ANSWER`, the verdict short-circuits

to CANNOT_ANSWER (Stage B is not invoked); otherwise, we run Stage B on (q, p, a^+) to obtain a verdict in {CORRECT, WRONG, CANNOT_ANSWER}. We keep the (q, p) pair iff the verdict is CORRECT, dropping it otherwise.

A.3.6 Training implementation details

Referenced from §4.2. We implement contrastive fine-tuning with GradCache to decouple the effective batch size from GPU memory. Training uses a batch size of 64 with a grad-cache chunk size of 4, 2 hard negatives per query, a peak learning rate of 7×10^{-6} with 20 warmup steps and cosine decay. The training set contains approximately 40K synthetic query–tile pairs after all filtering stages described above. Training completes in approximately 3 hours on a single H100 GPU.

A.4 Mini-Datstore for Embedding Ablations

Evaluating every embedding checkpoint against the full 30M-tile Wikipedia datstore is prohibitively expensive. We instead construct a compact mini-datstore for rapid iteration. We sample 400 queries from the evaluation set. For each query, the mini-datstore contains all tiles from the gold article plus the top-100 tiles retrieved by the base embedding model, producing a per-query pool that mixes relevant and irrelevant candidates. The resulting mini-datstore contains 400 questions and 7,426 tiles, approximating the difficulty of full-scale retrieval in a compact form suitable for rapid evaluation. All accuracy numbers reported in the embedding training recipe ablation (Table 4) are evaluated on this mini-datstore. We will publicly release this mini-datstore as a lightweight benchmark for evaluating visual retrieval over noisy webpage screenshots.

A.5 Reader Fine-tuning: Method and Results

Data. We reuse data from the embedding training pipeline (§4.1): each Stage-1 (q, p) pair already comes with a verified answer a from the answerability filter (Appendix A.3.2), giving us SFT triples (q, p, a) at no additional annotation cost. To simulate the retrieval setting, we use the embedding model from §4.2 to retrieve $k \in \{1, \dots, 6\}$ tiles for q over the datstore, and add p to the set if it is not already present. All tiles are downsampled via Lanczos resampling [59] to a target low resolution, where a $c \times$ compression scales each side by $1/\sqrt{c}$ (e.g., $4 \times$ halves both width and height). By construction, this data format teaches the reader to (i) read low-resolution images and (ii) ignore distractor tiles.

SFT loss. We fine-tune the reader ϕ with standard token-level cross-entropy over answer tokens: $\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(q, \mathcal{T}, a)} \sum_t \log P_\phi(a_t | \mathcal{T}, q, a_{<t})$, where \mathcal{T} is the retrieved tile set.

Evaluation. Table 6 reports LLM-judge accuracy (GPT-4.1) on a held-out 500-example test set, sweeping the compression factor c and the number of retrieved tiles k . At each c we compare the no-SFT base reader applied directly to compressed tiles (*compression-only*) against the SFT-trained reader, with the uncompressed base reader at $1 \times$ as the ceiling reference.

Training observations. (i) *Compression alone hurts.* Applying $2 \times$ compression to the base reader without retraining drops average accuracy from 0.905 to 0.854 (-5.1pp), and $3 \times$ drops it to 0.738 (-16.7pp). (ii) *SFT recovers the loss and meets or exceeds the uncompressed ceiling.* SFT at $2 \times$ reaches an average of 0.947, $+9.3\text{pp}$ above the compression-only baseline at the same c and $+4.2\text{pp}$ above the uncompressed ceiling itself; SFT at $3 \times$ reaches 0.910, $+17.2\text{pp}$ above its compression-only baseline and matching the uncompressed ceiling on average ($+0.5\text{pp}$) while using one third of the pixel budget. (iii) *SFT improves robustness to distractor tiles as k grows.* The base reader at $1 \times$ degrades from 0.958 at $k=1$ to 0.856 at $k=4$ (-10.2pp) as more distractors are added; SFT at $2 \times$ stays within 1.4pp of its $k=2$ peak across the full k sweep, consistent with the “ignore distractor / focus on gold tile” training goal described above.

B Evaluation Protocol

B.1 Benchmark Details

Table 7 summarizes the evaluation benchmarks used in this paper.

Table 6: Reader SFT training evaluation across compression ratios (GPT-4.1 judge, 500-example held-out set).

Setting	$k=1$	$k=2$	$k=3$	$k=4$	avg
Base @ 1× (no SFT, ceiling)	0.958	0.912	0.892	0.856	0.905
Base @ 2× (compression only, no SFT)	0.908	0.862	0.852	0.794	0.854
SFT @ 2×	0.946	0.950	0.954	0.936	0.947
Base @ 3× (compression only, no SFT)	0.830	0.726	0.710	0.684	0.738
SFT @ 3×	0.904	0.918	0.932	0.884	0.910

Table 7: Benchmark details. All Wikipedia benchmarks query the same 30M-page datastore (§3.1); LiveVQA queries a separate news datastore (Appendix A.2).

Benchmark	Task family	n	Metric	Source
Natural Questions	Text-centric Wiki QA	1,000	GPT-4.1 judge	[30]
NQ-Tables	Text-centric Wiki QA	1,000	GPT-4.1 judge	[51]
SimpleQA	Text-centric Wiki QA	1,000	GPT-4.1 judge	[31]
MMSearch	Multimodal Wiki QA	300	GPT-4.1 judge	[33]
Encyclopedic VQA	Multimodal Wiki QA	749	GPT-4.1 judge	[52]
LiveVQA-2025	News VQA	6,632*	Accuracy	[32]

* CNN/BBC/AP News subset; the full LiveVQA dataset has 26,888 QA pairs — we exclude Forbes and Variety due to anti-bot protections that prevent reliable page capture.

Encyclopedic VQA subset. We evaluate on the *landmarks* subset (Google Landmarks v2 [67]) with *automatic* questions only ($n=749$). The iNaturalist subset is excluded due to missing query images in the official dataset release; the automatic question type is used as it is the largest category and contains naturally phrased questions generated from Wikipedia sections.

MMSearch subset. We evaluate on all 300 end-to-end examples. Of these, 171 include a query image and 129 are text-only.

LiveVQA setup. Each query consists of an editorial photo and a multiple-choice question; we jointly embed the photo and question text to retrieve from the news tile index (Appendix A.2). We evaluate on CNN, BBC, and AP News ($n=6,632$ QA pairs), excluding Forbes and Variety due to anti-bot protections that prevent reliable page capture.

MoNaCo setup. We evaluate on the full MoNaCo benchmark [56] (1,315 multi-hop Wikipedia QA questions) using a GPT-5 ReAct agent with a single `search` tool. We use GPT-5 as the agent controller because it exhibits stronger and more stable agentic behavior than current open-source alternatives, allowing us to better isolate the impact of the retrieval backend. Only the search backend varies across conditions; the agent loop, prompt, and reader (GPT-5) are identical. We compare four backends: PIXELRAG (pixel retrieval), Trafilatura text retrieval, Google via SerpApi [57], and DS-Serve [58] (open-source neural retrieval endpoint). The agent issues up to 20 search queries per question with top- $k=5$ results per query.

B.2 Grading Protocol

All Wikipedia QA benchmarks (NQ, NQ-Tables, SimpleQA, MMSearch, Encyclopedic VQA) use LLM-as-judge grading with GPT-4.1 [68] at temperature 0, seed 42, and `max_tokens=1000`. Following [31], the grader classifies each prediction as CORRECT, INCORRECT, or NOT_ATTEMPTED; we score CORRECT as 1.0 and the rest as 0.0. For NQ and NQ-Tables, up to 10 gold answer aliases are joined with “OR” so any match counts as correct. LiveVQA is a 5-option multiple-choice task graded by exact letter match (no LLM grader). Reader prompt templates are listed in Appendix F.

C Additional Results

C.1 Results with Alternative Readers

The main text reports results with the Qwen3.5-4B reader. Tables 8–9 repeat the main results at top- $k=3$ and top- $k=1$ with the Qwen3-VL-4B reader. All directional findings are identical across reader variants. The tuned retriever (LoRA) consistently improves over the base checkpoint: on SimpleQA with the VL-4B reader, accuracy rises from 70.3% to 75.1% at top-3 and from 58.3% to 63.7% at top-1.

Table 8: End-to-end results with Qwen3-VL-4B reader (cf. Table 1).

Method	NQ		NQ-Tables		SimpleQA		MMSearch	EVQA		LiveVQA	
	Recall	Acc	Recall	Acc	Recall	Acc	Acc	Recall	Acc	Recall	Acc
No retrieval		13.1		10.9		6.4	13.5		27.3		54.2
<i>Text-based retrieval</i>											
mwparsersfromhell	48.6	27.7	35.0	18.3	72.9	58.4	12.9	3.2	30.7		
Trafilatura	45.8	29.4	37.7	21.9	79.3	69.0	13.5	3.2	29.2	16.2	55.9
<i>Pixel-based retrieval</i>											
PIXELRAG (base)	53.4	31.0	44.4	24.3	79.9	70.3	19.9	17.0	39.4	38.9	66.1
PIXELRAG	58.9	31.1	50.7	25.8	83.1	75.1	21.6	20.2	40.8	33.3	65.3

Table 9: End-to-end results at top- $k=1$ with Qwen3-VL-4B reader (cf. Table 1).

Method	NQ		NQ-Tables		SimpleQA		MMSearch	EVQA		LiveVQA	
	Recall	Acc	Recall	Acc	Recall	Acc	Acc	Recall	Acc	Recall	Acc
No retrieval		13.1		10.9		6.4	12.7		27.3		54.2
<i>Text-based retrieval</i>											
mwparsersfromhell	28.1	23.1	17.8	14.9	58.8	48.6	18.1	1.77	29.2		
Trafilatura	26.3	23.6	19.3	15.2	62.6	55.6	19.0	1.7	28.0	9.8	53.0
<i>Pixel-based retrieval</i>											
PIXELRAG (base)	36.3	25.7	28.2	20.0	65.7	58.3	20.3	9.8	37.0	26.8	61.2
PIXELRAG	40.3	27.1	33.7	21.7	71.2	63.7	22.7	12.9	39.4	22.4	60.8

Table 10: Retrieval–reader modality ablation with Qwen3-VL-4B reader (cf. Table 3).

Retrieval	Reader	SimpleQA	LiveVQA
Screenshot	Screenshot	70.3	66.1
Screenshot	OCR text	69.8	63.9
Text	Rendered img	63.1	56.8
Text	Text	69.0	55.9

C.2 Full Results of Scaling VLM Performance

Figure 7 visualizes the accuracy and token-cost trends from the full 31-model sweep in Table 5 (main text). Reasoning-mode models (bottom rows of the table) show a similar pattern but with higher output token cost due to chain-of-thought generation; the pixel advantage persists (+4.8 pp for Qwen3.5-4B reasoning, +4.6 pp for Qwen3.6-35B-A3B reasoning).

D Analysis

D.1 Wikipedia Text Extractor Comparison

To validate our choice of HTML-to-text extractor, we compare seven mainstream Wikipedia text parsers using GPT-4.1 as a judge on a six-dimension rubric covering QA-usability, completeness,

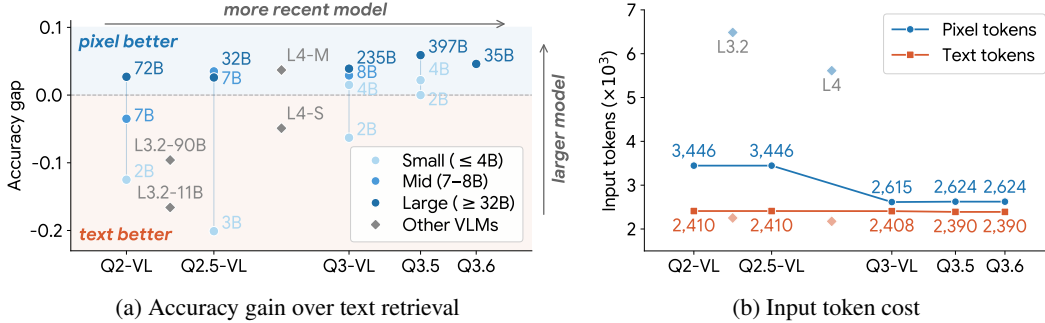


Figure 7: VLM reading ability across model generations on SimpleQA. Q = Qwen; L3.2 = Llama-3.2-Vision; L4 = Llama-4. Markers denote parameter scale.

cleanliness, tabular data, hierarchy, and overall quality, evaluated over 50 diverse Wikipedia pages (Table 11). `trafilatura` with `output_format='markdown'` produces the highest-quality output overall (6.40/10, winning 36 out of 50 pages), followed closely by `resiliparse` (6.34/10, 31 wins). Our pipeline uses `trafilatura` as the text-side parser for all text-retrieval baselines, **ensuring that the text backend we compare against is the strongest available.**

Critically, *even the best parser scores only 4.38/10 on tabular data*, and no parser exceeds 4.38 on this dimension. This ceiling is structural: linearization cannot preserve multi-row, multi-column alignment, merged cells, or the spatial grouping that makes table content interpretable. This observation directly motivates our pixel-space approach: for table-bearing content, bypassing the parser entirely and operating on rendered screenshots avoids the information loss that every text extractor introduces.

Table 11: Wikipedia text extractor comparison. GPT-4.1 judges seven parsers on six dimensions (scale 1–10) over 50 diverse pages; “Wins” counts pages where the parser ranked first overall. Bold marks column-best values. Even the top parser scores below 4.4/10 on tabular data, motivating pixel-space retrieval for table-bearing content.

Rank	Parser	Overall	QA-usab.	Compl.	Clean.	Tables	Hier.	Wins
1	<code>trafilatura</code> (md)	6.40	6.56	7.46	6.24	4.38	8.04	36
2	<code>resiliparse</code>	6.34	6.64	7.60	6.02	4.02	7.16	31
3	<code>BeautifulSoup</code>	6.06	6.26	7.20	6.88	3.86	5.88	29
4	<code>mwpaserfromhell</code>	5.56	5.60	6.52	6.24	3.16	6.60	15
5	<code>wikiextractor</code>	5.20	5.26	6.10	7.54	2.82	5.26	8
6	<code>raw wikitext</code>	4.60	4.92	6.12	3.62	2.76	5.86	3
7	<code>jina_reader</code>	2.30	1.78	1.84	2.22	1.80	4.90	2

D.2 Visual Information Loss During HTML Parsing

Figure 8 shows two concrete examples of visual information loss when HTML is parsed into plain text. In the first example, a match page containing formation diagrams, penalty shoot-out icons, and lineup tables is reduced to a near-empty string after parsing. In the second, a multi-row awards table with merged cells is flattened into a wall of text; for instance, December’s “Manager of the Month” is unrecoverable from the linearized output because the merged cell spans two rows. An additional end-to-end example from our evaluation appears in Figure 10 (§D.4): the 2010 Champions League Final article’s match-statistics table is destroyed by linearization, so no text chunk in the corpus contains the answer, whereas the pixel retriever surfaces the rendered table directly.

D.3 Retrieval Signal Loss Under Text Linearization

Figure 9 shows a representative MoNaCo retrieval trace for the question: “What is the current percentage of Indigenous peoples in each country in America?” The text retriever repeatedly surfaced passages from the correct topic area (an article lead, see-also links, and portal/navigation fragments) but none contained the country-by-country percentage table. The pixel retriever instead surfaced a mid-page table tile where the country rows and percentage values are visually grouped. This is not a case where extraction deleted the answer; rather, linearization made the answer-bearing table a weaker



Figure 8: Visual information loss during HTML parsing. Each row shows a rendered Wikipedia page (left) and the text extracted by `trafilatura` (right). Top: a match page with rich visual structure is reduced to a near-empty string. Bottom: a table with merged cells is linearized, making certain cell associations unrecoverable.

retrieval target than topic-matching prose and navigation text. In the logged run, no answer-bearing percentage chunk entered the text top-5 across the agent’s 12 queries. For the closest table-formulated text query, the corresponding same-article table/reference chunk appeared only at rank 38 (score 0.580), while the pixel query retrieved the rendered table tile at rank 2 (score 0.616). Two additional evaluation examples illustrate the same rank-loss pattern on SimpleQA: in Figure 11, text retrieval ranks the Dalí infobox above the answer-bearing body paragraph (rank 12 vs. rank 3 for PIXELRAG); in Figure 12, the gap is even more extreme, with the answer paragraph falling to rank 66 under text retrieval while PIXELRAG recovers it at rank 3.

D.4 Detailed Failure Decomposition

Evidence verification. To distinguish genuine retrieval failures from borderline misses, we deliberately use a strong verifier (GPT-5.1) over a generous retrieval scope (top-100): if no answer-bearing item appears anywhere in the top-100, the miss is structural rather than a ranking threshold artifact. Concretely, for each SimpleQA question we retrieve the top-100 text chunks and screenshot tiles from their respective indexes and ask GPT-5.1 whether each item contains enough information to answer the question (screenshot items use multimodal input; text items use text-only input). An item is *valid evidence* if GPT-5.1 answers correctly, verified by GPT-4.1 grading against the ground-truth answer. This two-stage verify-then-grade protocol mirrors the consistency-judge pipeline used for hard-negative mining during training (Appendix A.3.4): a capable model attempts to answer from the candidate, and a separate grader confirms correctness.

Evidence type classification. Each valid evidence *tile* is classified by the DOM structure of the chunk it occupies. We re-render the gold article in Playwright at the same 875px viewport used for tile

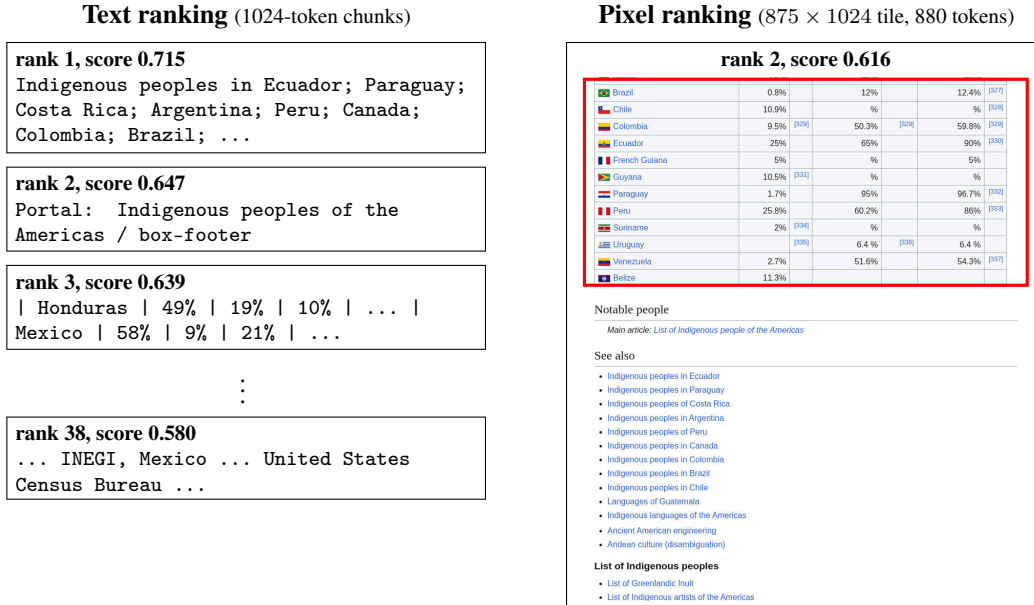


Figure 9: Retrieval signal loss under text linearization. The left panel shows ranked text chunks for the table-formulated query, with “...” marking omitted text. Across 12 text queries, the answer-bearing percentage chunks never entered the top-5, whereas the pixel path retrieved the rendered table tile containing the answer evidence.

capture, then for each 1024px vertical strip, query all DOM elements via `getBoundingClientRect` and assign the tile’s type as the element with the largest pixel coverage: `<table class="infobox">` → `infobox`, `<table>` → `table`, `` → `list`, `<p>` → `paragraph`. A question can have valid evidence tiles of multiple types; rather than projecting it onto a single bucket, we report each per-type analysis on the conditional sub-population of questions for which the corresponding type- τ valid evidence exists. The sub-populations therefore overlap, and per-type sample sizes do not sum to N .

Failure decomposition. Table 12 pairs each question’s text and screenshot outcomes. A question exhibits *parser loss* when PIXELRAG retrieval surfaces valid evidence in top-3 but no valid text chunk exists anywhere in the retrieved top-100, meaning the text corpus cannot answer the question. *Rank loss* applies when valid text evidence exists but falls outside the reader’s top-3 window. The rank-loss columns compare the two modalities on the rank-loss subset only: text fails by definition (rank > 3), while the PIXELRAG column shows where it succeeds on the same questions (rank ≤ 3 by definition).

Table 12: Paired failure decomposition on SimpleQA by evidence type, restricted to questions where PIXELRAG succeeds but text fails. Parser and Rank give the counts of parser-loss and rank-loss; the rank-loss columns give the mean evidence rank per modality on the rank-loss subset. Rows overlap (column sums > Overall).

Type	n	Text fail [†]		Rank-loss rank			Both fail
		Parser	Rank	PIXELRAG	Trafilatura	PIXELRAG fail	
Table	282	21	33	1.2	17.7	17	36
List	290	16	29	1.6	16.3	22	25
Infobox	503	29	41	1.4	16.6	24	36
Paragraph	571	34	73	1.6	22.5	30	58
Overall	946	67	91	1.5	20.1	55	99

[†]PIXELRAG succeeds (rank ≤ 3) but text does not.

Table 13 shows representative SimpleQA examples for each failure mode.

Table 13: Representative SimpleQA examples for each failure mode.

Mode	Ev. type	Question (abbreviated)	Answer	PIXELRAG	Trafilatura
Parser	table	How many shots did Inter attempt on target in the CL Final ... ?	7	1	>100
Parser	para.	What disease was Elizabeth Esteve-Coll diagnosed with ... ?	multiple sclerosis	1	>100
Parser	list	Which award did Reza Aslan receive in 2014?	Intersections Int’l Award	2	>100
Rank	table	On what day was Javier Zanetti’s first daughter born?	11 June 2005	2	89
Rank	para.	What was the rate of climb of the Grumman F4F-3 Wildcat ... ?	11.70 m/s	2	91
Rank	list	Who sketched the Taddei Tondo following its arrival at the RA ... ?	John Constable	1	62
Rank	infobox	What team finished with 38 pts in the 2021–22 PL season?	Leeds United	1	68

Reader loss. Reader loss is computed by cross-referencing the evidence annotations with actual reader outputs: questions where valid evidence appears in the reader’s top-3 but the reader still answers incorrectly. With the Qwen3.5-4B reader, reader loss is 7.0% (56 of 799 questions with valid PIXELRAG evidence in top-3).

Figures 10–13 show fully expanded case studies (one for parser loss, two for rank loss, one for reader loss), including the actual top-3 retrieved chunks and tiles for each modality and the resulting reader outputs.

Infobox rank displacement. Among the 91 rank-loss cases in Table 12, we find that 44% share a common pattern: text retrieval ranks the article’s infobox chunk (chunk 0) above the answer-bearing content despite the infobox not containing the answer. Wikipedia infoboxes contain the entity name, key attributes, and category labels, producing high keyword overlap with factual queries. Text retrieval systematically places chunk 0 at rank 1 for 75.9% of queries (vs. 67.1% for PIXELRAG retrieval), because the linearized infobox is a dense text-similarity target. PIXELRAG retrieval is less susceptible: the visual embedding captures the distinct layout of an infobox (bordered sidebar with key-value pairs) versus a body paragraph (flowing text under a section heading), allowing the retriever to distinguish content type even when keyword overlap is similar.

The effect is counter-intuitive because it is worst on the *correct article*: the retriever finds the right Wikipedia page and places its infobox at rank 1, yet the answer-bearing paragraph or table falls to rank 20+ because the infobox already saturates the top positions.

Examples include:

- **“What President nominated Elliott Fitch Shepard as U.S. Attorney?”** (answer: Rutherford B. Hayes). Text retrieval ranks the Shepard infobox chunk first (correct article, wrong section), while the answer-bearing paragraph falls to rank 66; PIXELRAG retrieves it at rank 3.
- **“What day did Dalí’s mother die?”** (answer: 6 February 1921). The Dalí infobox chunk is ranked first by text retrieval, but the answer appears in a body paragraph at rank 12; PIXELRAG retrieves it at rank 3.
- **“From which Israeli university did Judith Hemmendinger receive her master’s degree?”** (answer: Bar-Ilan University). Text retrieval places the infobox at rank 1, while the answer paragraph falls to rank 44; PIXELRAG retrieves it at rank 2.

The displacement effect is worst for paragraph evidence, where the mean rank of rank-loss cases is 22.5 (vs. 16–18 for other evidence types). In these cases, PIXELRAG retrieval finds the answer at mean rank 1.8 on the same questions. On the 42 questions where paragraph evidence ranks outside text retrieval’s top-3 *and* text places an infobox chunk at rank 1, PIXELRAG still recovers valid evidence in top-3 for 37 of 42 (88%, paragraph evidence specifically for 20 of 42), directly confirming its immunity to the displacement described above.

Q: “How many shots did Inter attempt on target in the Champions League Final ... May 23, 2010?” A: 7

Trafilatura text top-3

rank 1, score 0.694 — 2010 UEFA Champions League final (chunk 0)
 2010 UEFA Champions League final | Date | 22 May 2010 | Venue | Santiago Bernabéu, Madrid | ...
 ↪ *Infobox metadata; no shots-on-target stats.*

rank 2, score 0.670 — 2009–10 UEFA Champions League (chunk 0)
 2009–10 UEFA Champions League | Dates | 30 June ... 22 May 2010 | Champion | Inter Milan | ...
 ↪ *Season overview infobox, not match stats.*

rank 3, score 0.667 — Wesley Sneijder (chunk 13)
 "Bayern Munich 0-2 Internazionale". ESPN Soccer.net. 22 May 2010. Archived from ...
 ↪ *Bibliography, not data.*

rank 26, score 0.614 — 2010 UEFA Champions League final (chunk 4)
 Statistics | | | Post-match As a result of Inter's victory, Italy held onto its position ...

↪ *Statistics table destroyed by linearization; header survived, table body reduced to empty pipes.*

Text reader: "... the number of shots Inter attempted on target is not mentioned. ..." ✗

PIXELRAG screenshot top-3

rank 1, score 0.601 — 2010 UEFA CL final (tile 7)


Marin Åhansson (England)
 Reserve assistant referee:
 Peter Kirkup (England)^[7]

Statistics

Statistic	First half ^[4]		Second half ^[4]		Overall ^[4]	
	Bayern Munich	Inter Milan	Bayern Munich	Inter Milan	Bayern Munich	Inter Milan
Goals scored	0	1	0	1	0	2
Total shots	10	7	11	4	21	11
Shots on target	1	4	5	3	6	7
Saves	3	1	2	5	5	6
Ball possession	67%	33%	69%	31%	68%	32%
Corner kicks	2	0	4	2	6	2
Fouls committed	8	6	8	7	16	13
Offsides	0	0	0	0	0	0
Yellow cards	1	1	1	0	2	1
Red cards	0	0	0	0	0	0

Post-match

As a result of Inter's victory, Italy held onto its position in the top three of the UEFA country coefficient rankings and would therefore retain its fourth berth in the 2010–11 UEFA Champions League. If Bayern had won or the match had gone to penalties, Germany would have overtaken Italy and received an extra entry in 2010–11.^[6] Having beaten Roma both in the 2010 Coppa Italia Final and to the 2009–10 Serie A title, Inter won the treble for the first time in Italian football history.^[4]



Match winner Diego Milito's jersey and memorabilia from the final displayed at the San Siro museum

Despite winning the treble, José Mourinho resigned as manager of Inter on 28 May to take over from Manuel Pellegrini as manager of Real Madrid, citing a desire to become the first manager to win the Champions League with three clubs.^[4] However, he was unable to accomplish this feat, being eliminated in

Pixel reader: "... under the 'Overall' column for 'Inter Milan', the number of 'Shots on target' is 7." ✓

Figure 10: **Parser loss.** The 2010 Champions League Final article’s match-statistics table is destroyed by HTML-to-text linearization, so no text chunk in the corpus contains the answer. The pixel retriever surfaces the rendered statistics table as the top tile. This is the same example shown in Figure 1; here we expand the full retrieval lists for both modalities.

Table 14 reports, for each modality, the share of its failures falling into each of the three modes (*parser loss*: no answer-bearing item anywhere in top-100; *rank loss*: retrieved but outside top-3; *reader loss*: in top-3 but the reader still errs). The headline text percentages in §5.2 (36.6%/55.2%/8.2%) are the question-level aggregate of these columns over all evidence types. Table 15 reports the rank distribution of valid type- τ evidence in top-100 retrieval, indicating how many evidence misses at $k=3$ are near misses recoverable at larger k .

Table 14: SimpleQA failure causes by evidence type for text vs. PIXELRAG retrieval (Qwen3.5-4B reader, top- $k=3$). Parser/Rank/Reader give the % of each modality’s failures from parser, rank, and reader loss. Rows overlap.

Modality	Evidence	n	Failures	Parser	Rank	Reader	Ev. Recall@3
PIXELRAG	Table	282	78	6.4	56.4	37.2	81.2
PIXELRAG	List	290	73	8.2	50.7	41.1	83.8
PIXELRAG	Infobox	503	92	5.4	55.4	39.1	88.1
PIXELRAG	Paragraph	571	121	4.1	65.3	30.6	84.6
Text	Table	282	95	35.8	49.5	14.7	68.1
Text	List	290	70	31.4	58.6	10.0	75.9
Text	Infobox	503	111	28.8	57.7	13.5	78.9
Text	Paragraph	571	167	29.3	59.9	10.8	71.1

D.5 HTML DOM Lookup Baseline: Setup and Analysis

The **Text** → **HTML** row in Table 3 tests whether preserving the original DOM structure of Wikipedia articles can close the gap between text-based RAG and pixel-based RAG. Standard text-based RAG

Q: "What day, month, and year did Salvador Dalí's mother pass away?" A: 6 February 1921

Trafilatura text top-3

rank 1, score 0.662 — *Salvador Dalí* (chunk 0)
 Salvador Dalí | Born | 11 May 1904 | Died | 23 January 1989 | Education | San Fernando ...

→ *Infobox; lists Dalí's own dates, not his mother's.*

rank 2, score 0.649 — *Gala Dalí* (chunk 1)
 ...Death. Gala died in Port Lligat in Catalonia, Spain, earl...

→ *Wrong person (Dalí's wife, not his mother).*

rank 3, score 0.639 — *Salvador Dalí* (chunk 20)
 at archive.today -- Boletín Oficial del Estado ... Gibson, Ian (1997) pp. 603-604 ...

→ *References section, no biographical content.*

rank 12 — *Salvador Dalí* (chunk 1)
 ...On 6 February 1921, Dalí's mother died of uterine cancer. Dalí was 16 years old and later said his mother's death "was the greatest blow I had experienced in my life" ...

PIXELRAG screenshot top-3

rank 1, score 0.566 — *Salvador Dalí* (tile 0) **rank 2, score 0.565** — *Salvador Dalí* (tile 1)

rank 3, score 0.563 — *Salvador Dalí* (tile 2: contains "6 February 1921")

Dalí also had a sister, Ana María, who was three years younger,^[13] and whom Dalí painted 12 times between 1923 and 1926.^[22]

His childhood friends included future FC Barcelona footballers **Emili Sagüés** and **Josap Samitier**. During holidays at the Catalan resort town of **Cadaqués**, the trio played football together.^[23]

Dalí attended the Municipal Drawing School at Figueras in 1916 and also discovered modern painting on a summer vacation trip to Cadaqués with the family of **Ramon Pichot**, a local artist who made regular trips to Paris.^[24] The next year, Dalí's father organized an exhibition of his charcoal drawings in their family home. He had his first public exhibition at the Municipal Theatre in Figueras in 1918,^[24] a site he would return to decades later. In early 1921 the Pichot family introduced Dalí to **Futurism**. That same year, Dalí's uncle Anselm Domènech, who owned a bookshop in Barcelona, supplied him with books and magazines on **Cubism** and contemporary art.^[25]

On 6 February 1921, Dalí's mother died of uterine cancer.^[26] Dalí was 16 years old and later said his mother's death "was the greatest blow I had experienced in my life. I worshipped her ... I could not resign myself to the loss of a being on whom I counted to make invisible the unavoidable blemishes of my soul."^[8:27] After the death of Dalí's mother, Dalí's father married her sister. Dalí did not resent this marriage, because he had great love and respect for his aunt.^[13]

Madrid, Barcelona and Paris

In 1922, Dalí moved into the *Residencia de Estudiantes* (Students' Residence) in Madrid^[27] and studied at the *Real Academia de Bellas Artes de San Fernando* (San Fernando Royal Academy of Fine Arts). A lean 1.72 metres (5 ft 7½ in) tall,^[28] Dalí already drew attention as an eccentric and dandy. He had long hair and sideburns, coat, stockings, and knee-breeches in the style of English aesthetes of the late 19th century.^[29]

At the Residencia, he became close friends with **Pepín Betis**, **Luis Buñuel**, **Federico García Lorca**, and others associated with the Madrid avant-garde group **Ultra**.^[30] The friendship with Lorca had a strong element of mutual passion,^[31] but Dalí said he rejected the poet's sexual advances.^[32] Dalí's friendship with Lorca was to remain one of his most emotionally intense relationships until the poet's death at the hands of **Nationalist** forces in 1936 at the beginning of the **Spanish Civil War**.^[7]

Also in 1922, he began what would become a lifelong relationship with the **Prado Museum**, which he felt was, 'incontestably the best museum of old paintings in the world.'^[33] Each Sunday morning, Dalí went to the Prado to study the works of the great masters. 'This was the start of a monk-like period for me, devoted entirely to solitary work: visits

Text reader: "... the specific day and month of Salvador Dalí's mother's death are not mentioned; only the year 1921 is given." ✗

Pixel reader: "... Salvador Dalí's mother passed away on 6 February 1921." ✓

Figure 11: **Rank loss (paragraph evidence).** Once the infobox is linearized, its flattened key–value text out-ranks the answer-bearing body paragraph (which falls to rank 12) — the infobox lists Dalí's own birth/death, not his mother's, yet matches the query on the entity name. The visual embedding keeps the infobox sidebar structurally distinct from the body section, surfacing the relevant tile in the top-3.

Table 15: SimpleQA evidence-tile rank distribution by evidence type (top-100 retrieval, Qwen3.5-4B reader). Rows overlap.

Modality	Evidence type	<i>n</i>	Recall@3	Recall@10	Recall@50	Median rank
PIXELRAG	Table	282	34.8	51.8	77.7	7
PIXELRAG	List	290	36.9	52.1	76.9	6
PIXELRAG	Infobox	503	63.0	69.2	77.7	1
PIXELRAG	Paragraph	571	63.9	76.9	87.2	2
Text	Table	282	23.8	34.0	45.7	4
Text	List	290	28.6	33.4	44.5	2
Text	Infobox	503	65.2	71.6	77.9	1
Text	Paragraph	571	44.1	54.6	66.4	2

Q: “What President nominated Elliott Fitch Shepard as U.S. Attorney for the Southern District of New York?”
A: **Rutherford B. Hayes**

Trafilatura text top-3

rank 1, score 0.689 — *Elliott Fitch Shepard* (chunk 0)
 Elliott Fitch Shepard | Died | March 24, 1893 New York City | Occupation | lawyer, banker ...
 → *Infobox; does not list the nominating President.*

rank 2, score 0.668 — *U.S. Attorney for the District of New York* (chunk 0)
 The U.S. Attorney for the District of New York was from 1789 to 1815 the chief federal law ...
 → *Office history page, no per-appointee details.*

rank 3, score 0.658 — *U.S. Attorney for the Southern District of New York* (chunk 0)
 Formed | September 24, 1789 Judiciary Act of 1789 | Jurisdiction | Southern District ...
 → *Office overview infobox, no list of nominees.*

rank 66 — *Elliott Fitch Shepard* (chunk 1)
 ... In 1881, US President Rutherford B. Hayes nominated him for United States Attorney for the Southern District of New York ...

Text reader: “... there is no information stating that Elliott Fitch Shepard was nominated as United States Attorney ...” ✗

PIXELRAG screenshot top-3

rank 1, score 0.576 — *Elliott Fitch Shepard* (tile 0) **rank 2, score 0.512** — *Elliott Shepard* (tile 0)

rank 3, score 0.509 — *Elliott Fitch Shepard* (tile 2; answer region boxed)

2024^[19],^[20] He became one of the school's trustees and vice presidents.^[19]

In 1868, Shepard became a partner of Judge **Theron R. Strong** in Strong & Shepard, continuing the business after Strong's death.^[72] He continued to practice law for the next 25 years,^[6] he helped found the New York State Bar Association in 1876, and in 1884 was its fifth president.^{[22][23]} In 1875 Shepard drafted an amendment establishing an arbitration court for the **New York Chamber of Commerce**, serving on its five-member executive committee the following year.^[24] In 1880, the **New York City Board of Aldermen** appointed Shepard and Ebenezer B. Shaffer to revise and codify the city's local ordinances to form the New-York Municipal Code; the last revision was in 1896.^[725]

During the 1880s he helped found three banks. At the Bank of the Metropolis, he was a founding board member.^[27] The others were the American Savings Bank and the Columbian National Bank, where he served as chairman.^[28] In 1881, US President Rutherford B. Hayes nominated him for United States Attorney for the Southern District of New York.^[6] In 1884, Shepard led the effort to create an arbitration court for the **New York Chamber of Commerce**.^[29] On **Waterloo, 1866**, Shepard purchased the *small and obscure newspaper* (founded in 1836, with an estimated value in 1888 of \$200,000 (\$7 million in 2024^[18]) from **Cyrus W. Field**^[22] for \$425,000 (\$14.9 million in 2024^[18],^[28]) Deeply religious, Shepard placed a verse from the Bible at the head of each edition's editorial page. As president of the newspaper company until his death, he approved every important decision or policy.^[29] In the same year, Shepard became the controlling stockholder of the **Fifth Avenue Transportation Company** to force it to halt work on Sundays (the **Christian Sabbath**).^[30]

When Margaret's father died in 1885, she inherited \$12 million (\$420 million in 2024^[18],^[6]) The family lived at 2 West 52nd Street in Manhattan,^[31] one of three houses of the **Vanderbilt Triple Palace** which were built during the 1880s for William Henry Vanderbilt and his two daughters. After Elliott's death, Margaret transferred the house to her sister's family, who combined their two houses into one.^[32] The houses were eventually demolished; the nine-story **De Forest Building** was built there in 1928 and was demolished around 1969.^[33] **650 Fifth Avenue** is the building currently on the site.

Shepard and his family toured the world in 1884,^[73]^[14] visiting Asia, Africa, and Europe.^[7] He documented his 1887 trip from New York to Alaska in *The River, New York and Alaska* taken by himself, his wife and daughter, six other family members, their maid, a chef, butler, *porter* and conductor. According to Shepard, the family traveled 14,085 miles (22,668 km) on 26 railroads and stayed at 38 hotels in nearly five months.^{[34][35]} After the 1884 trip, aware of the opportunity for church work in the territory, he founded a mission and maintained it with his wife for about \$20,000 (\$699,900 in 2024^[18]) a year. For some time Shepard worshipped at the **Fifth Avenue Presbyterian Church** under John Hall,^[3] and was a vice president of the Presbyterian Union of New-York.^[6] Shepard was president of the American Sabbath Union for five years,^[13] and he also served as the chairman of the Special Committee on Sabbath Observance.^[36]

Briarcliff Manor developments

Pixel reader: “... US President Rutherford B. Hayes nominated Elliott Fitch Shepard ... for the Southern District of New York in 1881.” ✓

Figure 12: **Rank loss (extreme rank gap).** Text retrieval places the Shepard infobox chunk at rank 1 — the correct article, but the infobox does not list the nominating President. The body paragraph that does contain the answer falls all the way to rank 66. The pixel retriever surfaces the answer-bearing tile at rank 3.

Q: “Who received an honorable mention at the 1996 Frank and Brennie Morgan Prize ... ?” A: **Lenhard Ng**

Trafilatura text top-3

```
rank 1 — Morgan Prize (chunk 0)
...Previous winners
- 1995
- Winner: Kannan Soundararajan ...
- Honorable mention: Kiran Kedlaya (Harvard)
- 1996
- Winner: Manjul Bhargava ...
- Honorable mention: Lenhard Ng (Harvard)
- 1997
- Winner: Jade Vinson ...
[...20 more year-name entries ...]
```

⇒ Correct answer present but buried in a flat list spanning 25 years; year–role–name hierarchy flattened to uniform dashes.

⋮

PIXELRAG screenshot top-3

```
rank 1, score 0.629 — Morgan Prize (tile 0)
Morgan Prize
Not to be confused with De Morgan Medal.
The Morgan Prize (full name Frank and Brennie Morgan Prize for Outstanding Research in Mathematics by an Undergraduate Student) is an annual award given to an undergraduate student in the US, Canada, or Mexico who demonstrates superior mathematics research. The $1,200 award, endowed by Mrs. Frank Morgan of Allentown, Pennsylvania, was founded in 1995. The award is made jointly by the American Mathematical Society, the Mathematical Association of America, and the Society for Industrial and Applied Mathematics. The Morgan Prize has been described as the highest honor given to an undergraduate in mathematics. [1]
Previous winners
1995
Winner: Kannan Soundararajan (Analytic Number Theory, University of Michigan)
Honorable mention: Kiran Kedlaya (Harvard University)
1996
Winner: Manjul Bhargava (Algebra, Harvard University)
Honorable mention: Lenhard Ng (Harvard University)
1997
Winner: Jade Vinson (Analysis and Geometry, Washington University in St. Louis)
Honorable mention: Vikas S. Sohail (Harvard University)
1998
Winner: Daniel Biss (Combinatorial Group Theory and Topology, Harvard University)
Honorable mention: Aaron F. Archer (Harvey Mudd College)
1999
Winner: Sean McLaughlin (Proof of the Dodecahedral Conjecture, University of Michigan)
Honorable mention: Samit Dasgupta (Harvard University)
2000
Winner: Jacob Lurie (Lie Algebras, Harvard University)
Honorable mention: Wai Ling Yee (University of Waterloo)
2001
Winner: Ciprian Manolescu (Floer Homology, Harvard University)
Honorable mention: Michael Levin (Massachusetts Institute of Technology)
2002
Winner: Joshua Greene (Proof of the Kneser conjecture, Harvey Mudd College)[2]
Honorable mention: None
2003
Winner: Melanie Wood (Belyi-extending maps and P-orderings, Duke University)[3]
```

⇒ Visual grouping preserves year–role–name hierarchy; 1996 entry is unambiguous.

⋮

Text reader: “Kiran Kedlaya (Harvard University) received an honorable mention at the 1996 ... Morgan Prize.” ✗

⇒ Picks the 1995 honorable mention instead of 1996 — adjacent-entry confusion.

Pixel reader: “Lenhard Ng (Harvard University)” ✓

Figure 13: **Reader loss.** Both modalities retrieve the same gold article at rank 1, and the answer appears verbatim in the text chunk. However, the linearized list flattens the year–role–name hierarchy into uniform dash-prefixed lines, and the text reader attributes the 1995 honorable mention to 1996. The rendered tile preserves the visual grouping by year, allowing the VLM to locate the correct entry.

linearizes HTML into flat strings, destroying table and list structure; this baseline instead feeds the reader *raw HTML* with intact `<table>`, ``, and sectional markup.

Setup. We reuse the same Trafilatura text-chunk index and retrieval API (Qwen3-VL-Embedding-2B, 1024-token chunks, IVFFlat). For each retrieved text chunk, a *DOM lookup* step recovers the corresponding HTML from the original Wikipedia article:

1. **Fetch HTML.** The article HTML is served from a local Kiwix ZIM archive [69] via `kiwix-serve`, eliminating network latency.
2. **Extract search keys.** Distinctive phrases are extracted from the text chunk: table cell values (e.g., codes like B01AC06, numbers with units) for table-heavy chunks, mid-line prose fragments for paragraph-heavy chunks. The first line (article title) is skipped to avoid matching the `<h1>` heading.
3. **Locate in DOM.** Each key is searched within the `text_content()` of every element under the article’s `mw-parser-output` container. Both the key and element text are normalized (non-breaking spaces, dash variants, and diacritics are collapsed) to handle encoding mismatches between Trafilatura output and raw HTML. The tightest-matching element is selected.
4. **Resolve to contiguous span.** Each matched element is walked up to its nearest direct-child ancestor of `mw-parser-output`. The final result is the contiguous range of direct children from the first matched child to the last—preserving all intermediate elements (tables, paragraphs, lists) that the original text chunk spanned.

5. **Clean and return.** Inline `<style>`, `<script>`, and navigation-box (`navbox`) elements are stripped. The serialized HTML is returned to the reader. If no key matches in the DOM, the original flat text is used as fallback.

The reader (Qwen3-VL-4B, `max_model_len=65536`) receives the concatenated HTML of all $k=3$ retrieved passages, separated by `<hr>` delimiters.

Results. HTML achieves 59.8% QA accuracy vs. 71.6% for flat text on SimpleQA, and 56.6% vs. 59.0% on LiveVQA (Table 3). Retrieval quality is identical (Recall@1 and Recall@any differ by <1 pp), confirming that the gap is entirely in the reading stage. Oversized HTML passages are truncated to 30k characters per passage to keep the total context within the reader’s 65k-token window.

Why HTML hurts: tag dilution. HTML markup inflates the average context from 7,601 to 28,941 characters (3.8×) on SimpleQA, consuming reader tokens on tags rather than content. The reader sees the same factual content—but diluted by structural markup, it more frequently fails to locate the answer. The gap is smaller on LiveVQA (−2.4 pp vs. −11.8 pp on SimpleQA) because news articles are shorter and have simpler DOM structure than Wikipedia pages.

Implications. Structured HTML is semantically richer than linearized text, yet this richness comes at a steep token cost that current context-window–limited readers cannot absorb. Screenshots bypass both problems: a rendered tile encodes the same tabular and sectional structure in a fixed ~875 visual tokens per tile regardless of article complexity, and no markup overhead is paid. The HTML baseline thus supports the central claim of this paper: pixel-space retrieval preserves document structure without the linearization losses of text *or* the token overhead of markup.

D.6 Directly RAG on Raw HTML Data

One might expect that preserving the original HTML structure throughout the RAG pipeline would help, since linearization is lossy. We test this by building a fully HTML-native pipeline that indexes and reads raw HTML chunks directly, and find that it performs *worse* than plain-text RAG on nearly every benchmark: retrieval quality is comparable, but HTML markup overwhelms the reader with tags, causing large accuracy drops. We detail the setup and results below.

Setup. We extract raw HTML from the Kiwix ZIM archive and chunk at DOM boundaries: section headers (`<h2>/<h3>`) force chunk boundaries, prose elements accumulate up to 1,024 tokens, and tables are split at `<tr>` row boundaries with the header row prepended to each sub-chunk (preserving column context). Navigation boxes, reference lists, and table-of-contents elements are filtered out. Each chunk stores the raw HTML (with tags) and, for table sub-chunks, a `parent_html` field containing the full original table. The resulting corpus contains 25.7M chunks (vs. 15.7M for the Trafilatura 1,024-token text baseline).

Chunks are embedded with the same Qwen3-VL-Embedding-2B model used for all other baselines, with the raw HTML as input text (`max_length=1,024`). Short chunks are batched together via a dynamic token-budget scheduler to avoid padding waste. A FAISS IVFFlat index (`nlist=4,096`, `nprobe=128`) is built over the 25.7M embeddings. At query time, the search API returns the raw HTML chunk to the reader.

Table 16: HTML-RAG (full pipeline) vs. Trafilatura text baseline across Wikipedia benchmarks. Reader: Qwen3.5-4B, top- $k=3$, -no-think. Retrieval Recall@3 is computed over examples with ground-truth article annotations.

Method	NQ		NQ-Tables		SimpleQA		MMSearch	EVQA	
	R@3	Acc	R@3	Acc	R@3	Acc	Acc	R@3	Acc
Trafilatura (text)	45.8	55.9	37.2	42.5	76.2	69.2	24.7	6.4	29.6
HTML-RAG (ours)	47.9	26.5	38.2	18.4	69.6	59.8	22.3	1.5	34.7
Δ	+2.1	−29.4	+1.0	−24.1	−6.6	−9.4	−2.4	−4.9	+5.1

Results. Table 16 reveals a striking dissociation between retrieval and reading when HTML is used end-to-end:

- **Retrieval quality is comparable or better.** HTML-RAG achieves higher Recall@3 than Trafilatura on NQ (+2.1 pp) and NQ-Tables (+1.0 pp). DOM-boundary chunking produces more semantically coherent units than fixed-window text splitting, and the embedding model handles HTML markup without difficulty.
- **Reading quality drops sharply.** QA accuracy falls on every benchmark except EVQA, with the largest gaps on NQ (−29.4 pp) and NQ-Tables (−24.1 pp). The cause is the same as in the DOM lookup baseline (Appendix D.5): HTML tags inflate the reader’s context, consuming tokens on markup rather than evidence. The effect is most severe on knowledge-intensive benchmarks where the reader must locate a specific fact within a dense passage.
- **EVQA is the sole exception** (+5.1 pp). EVQA questions often target entity attributes found in structured infoboxes; the HTML preserves this structure, benefiting the reader even at higher token cost. However, the low retrieval recall (1.5% vs. 6.4%) suggests this gain is largely driven by the no-retrieval baseline rather than retrieved HTML content.

Implications. The HTML-RAG experiment reinforces the DOM lookup finding from a different angle: even when retrieval is built from scratch with HTML-native chunking and embedding, the reader bottleneck persists. The fundamental issue is not retrieval quality but *token efficiency*: HTML markup is a verbose encoding of structure that penalizes context-window-limited readers. Screenshots encode the same structural information in a fixed number of visual tokens (~875 per tile), avoiding both the linearization losses of text and the tag overhead of HTML. Raw HTML is therefore not a viable alternative to either text-based or pixel-based RAG. This underscores why text parsers exist in the first place: current LLM readers cannot consume raw HTML effectively, so careful extraction into clean text remains a necessary preprocessing step for text-based RAG, with all the associated information loss discussed in §2.

E Limitations, Broader Impact, and Future Work

Limitations. A key limitation of pixel-space retrieval is the loss of hyperlink structure. In text-based retrieval, hyperlinks provide navigable connections between documents and serve as a rich signal for downstream tasks such as multi-hop reasoning and entity disambiguation. In our screenshot-based representation, hyperlinks are visually rendered (e.g., as blue underlined text) but are not directly actionable; the system cannot follow a link to retrieve the target page. One mitigation is to preserve hyperlink information as structured metadata alongside each tile: by extracting hyperlink URLs and their anchor text during the rendering stage and storing them as auxiliary fields, downstream models can access link targets without requiring the pixel representation itself to encode this information.

A second limitation is storage overhead: storing rendered screenshots requires substantially more disk space than raw text. Our Wikipedia datastore alone occupies nearly 6 TB of screenshot tiles. A practical mitigation is a *render-on-demand* strategy: once all tiles have been embedded, the raw screenshots can be deleted, retaining only the embeddings for retrieval. At inference time, after retrieving the top- K tiles, the system looks up the corresponding HTML source and re-renders the relevant screenshots on the fly before passing them to the VLM for generation.

A third limitation is language coverage: all datastores in this work are English-only (English Wikipedia and English-language news outlets), introducing a language bias. Extending to multilingual corpora is an important direction for future work.

Broader impact. By operating directly on rendered screenshots, PIXELRAG removes the dependency on HTML parsing and text extraction that systematically disadvantages visually rich webpages (e.g., infographics, styled tables, diagram-heavy articles). This levels the playing field for content whose value is inseparable from its visual layout. The same principle extends beyond webpages: any visually rich document, such as scanned PDFs, slide decks, or posters, can be ingested as screenshot tiles and retrieved without format-specific parsers. Because pixel representations do not depend on language-specific tokenizers or extractors, this approach naturally extends to non-Latin-script and low-resource languages where text extraction pipelines are less mature. On the risk side, screenshot-based retrieval faithfully preserves whatever appears on a rendered page, including potentially harmful,

misleading, or private content. Unlike text pipelines, where filtering can operate on extracted strings, pixel content is harder to moderate automatically. Deploying PIXELRAG at scale therefore requires careful content filtering at the rendering or indexing stage to prevent surfacing inappropriate material.

Future work. Several directions can extend PIXELRAG. First, our embedding model is trained on a single domain (English Wikipedia); adapting it to a given target datastore, or mixing datastores from diverse domains (e.g., scientific papers, e-commerce, forums) to train a cross-domain embedding model, is a natural next step enabled by our synthetic training pipeline. Second, hybrid text and image retrieval, where text-based and pixel-based scores are combined, may capture complementary signals and further improve recall. Third, pixel-space datastores open new possibilities for agentic workflows: an agent could retrieve relevant screenshot tiles, visually ground its reasoning, and iteratively refine its search, leveraging the rich visual context that static text pipelines discard.

F Prompt Listings

This section collects the full verbatim prompts referenced throughout the appendix.

The figure shows a rendered Wikipedia tile on the left and a JSON record on the right. The tile contains text about the 1922 Rose Bowl game, including details about the teams (Washington and Jefferson), the players (Charles Fremont West and Hal Erickson), and the game's significance. The JSON record on the right is a structured output of a query and answer, with fields for 'query', 'answer', 'source_sentence', 'source_type', 'subject', 'title', and 'url'.

Model output → parsed record

```
{
  "query": "Who was the first African-American to play quarterback in the Rose Bowl?",
  "answer": "Charles Fremont West",
  "source_sentence": "Charles Fremont West was the first African-American to play quarterback, and Hal Erickson became the only man ever to play in two Rose Bowls, with two teams, without losing.",
  "source_type": "prose",
  "subject": "sports",
  "title": "1922 Rose Bowl",
  "url": "https://en.wikipedia.org/wiki/1922_Rose_Bowl"
}
```

Figure 14: Example of synthetic query generation (Stage 1). The rendered tile (left) is sent together with the prompt in Figure 15; the structured five-line output is parsed into the JSON record shown (right).

Synthetic query generation prompt

Image input

{screenshot tile}

You are generating a query-evidence pair for training a visual retrieval model over Wikipedia screenshot chunks.

TASK: Given this screenshot chunk, generate ONE factual question whose answer is explicitly and completely visible in this chunk.

STYLE -- write natural search-style questions, not templates.

Vary the phrasing: "how much", "in what year", "which", "who", "where", "what caused", "how long", etc. Examples of the target style:

- "How much money, in euros, was the surgeon held responsible for paying in the Olivia Puls case?"
- "In what city was the 2010 FIFA World Cup opening ceremony held?"
- "How many days did the 1906 San Francisco earthquake fire burn?"
- "Which award did Fullmetal Alchemist win at the American Anime Awards in 2007?"
- "Who was the first Black female judge appointed to the Cook County Circuit Court?"

EVIDENCE: draw from any visible content -- prose, infobox fields, table cells, image captions, diagrams, or photographs. Pick whichever source yields the most natural question; do not default to infobox.

HARD RULES:

1. **SELF-CONTAINED.** The question must be understandable on its own; every entity must be named explicitly.

BAD: "Who composed the music for the film?" (missing film name)

BAD: "On what date was Lerew awarded the DFC?" (surname only + acronym)

BAD: "Which cyclist placed second in the Tempo race?" (missing event/year)

BAD: "Which mission is shown in the screenshot?" (references layout)

GOOD: "Who composed the music for Once Upon a Time in Hong Kong?"

GOOD: "On what date was RAF pilot Arthur Lerew awarded the Distinguished Flying Cross in World War II?"

2. **EVIDENCE COMPLETE.** The answer must be fully visible in this chunk. The source span (S:) must be a complete, untruncated sentence.

3. **DISTINCTIVE.** Include enough specifics (names, dates, locations, titles) to distinguish this chunk from similar pages.

ANSWER: prefer a single concise entity -- name, date, place, number, title, or short phrase.

SKIP (write exactly: SKIP) if any of the following holds:

- Content is a raw vote count, track listing, census table, or episode list.
- The answer is not fully visible or requires external context.
- You cannot write a self-contained question naming every entity.
- The source sentence is truncated or a fragment.

OUTPUT -- exactly five lines, or the single literal word SKIP:

Q: <natural, self-contained question>

A: <concise answer>

S: <verbatim complete span from the chunk>

T: image | table | infobox | prose

C: science | medicine | history | geography | technology | education | culture | politics | economics | biology | sports | entertainment | other

Figure 15: Synthetic query generation prompt (Stage 1). The model is sent this text together with the rendered tile as an image in the same turn.

Self-contained-query filter prompt

For each numbered question, answer YES (self-contained) or NO (not self-contained).

A question is NOT self-contained (NO) if it requires knowing a specific Wikipedia page, table, or screenshot to understand WHAT is being asked. Specifically answer NO when:

1. The subject is a vague pronoun or generic noun without a proper name:
NO: "What was the final score of the basketball game between THE TEAM and Marquette?" ("the team" unnamed)
NO: "Who directed the episode of THE TELEVISION SERIES titled 'X'?" ("the television series" unnamed)
NO: "In what year did THE SUBJECT OF THE ARTICLE move to Tokyo?" ("the subject" unnamed)
NO: "What is the running time of THE FILM DESCRIBED IN THE TEXT?" (layout reference)
2. The question explicitly references document structure:
NO: "Which item IS LISTED IN THE TABLE as X?"
NO: "What is shown IN THE INFOBOX?"
NO: "According to THE PROVIDED TABLE, which..."
3. A role/position question where no year or identifying event is given and the role has had many holders:
NO: "Who was THE CAPTAIN of HMS Defence?" (no year, hundreds of captains over centuries)
4. A geographic entity refers only to a category without naming which one:
NO: "On what date did THE GOODS YARD at the London and North Eastern Railway station close?" (LNER had hundreds of stations -- which one?)

Answer YES if all the key entities (people, places, works, teams, events) are explicitly named, even if the names are obscure. Proper names are always fine.

- YES: "Who did Sandefjord Fotball hire as manager after firing Arne Sandsto?"
YES: "How many consonants does the Pesisir language have?"
YES: "What 'fresh' rating did the film Our Man in Havana receive on Rotten Tomatoes?"
YES: "In what city were the 2025 Special Olympics World Winter Games held?"
YES: "Who did Emile Derlin Zinsou serve as assistant to in 1945?"

Output exactly one line per question, using the question number: "1: YES" or "1: NO"

Questions:
{questions}

Figure 16: Self-contained-query filter prompt (Stage 1, first false-positive filter). Queries labelled NO are dropped from the training set.

Hard-negative Stage A: candidate-answer prompt (VLM)

You are looking at {tile_count} screenshot tiles from Wikipedia pages.

Based ONLY on what you can see in these images, answer the following question.

If the answer is not visible in the images, reply "CANNOT_ANSWER".

Question: {question}

Give a short, direct answer (just the answer, no explanation).

Hard-negative Stage B: judge prompt

You are validating a candidate answer against screenshot tiles from Wikipedia pages.

Based ONLY on what you can see in these images, classify the candidate answer to the question as exactly one of:

- CORRECT: the candidate answer is visible in the images and is correct.
- WRONG: the images contain enough information to tell that the candidate answer is wrong.
- CANNOT_ANSWER: the images do not contain enough information to verify the candidate answer.

Question: {question}

Candidate answer: {candidate_answer}

Return exactly one token: CORRECT, WRONG, or CANNOT_ANSWER.

Evidence QA prompt (text-only query)

You are a research assistant who answers questions based on provided evidence.

Use <think></think> tags to show your reasoning if needed.

Answer the question directly and concisely based ONLY on the provided evidence.

Evidence QA prompt (multimodal query)

You are a research assistant who answers questions based on retrieved visual evidence.

You will receive: (1) a text question, (2) a query image, and (3) retrieved Wikipedia evidence images.

Use the query image and evidence images to answer the question.

Use <think></think> tags to show your reasoning if needed.

Answer the question directly and concisely.

Figure 17: Evaluation prompts. **Hard-negative Stage A** (*answer*, blue): the VLM sees only the candidate tile(s) and the query, returning a short answer or CANNOT_ANSWER. **Stage B** (*judge*, blue): classifies the candidate as CORRECT (false negative, dropped), WRONG, or CANNOT_ANSWER (hard negative, kept). **Evidence QA** (gray): reader system prompts for text-only (top) and multimodal (bottom) query benchmarks.