PIXELRAG: RETRIEVAL AND GENERATION IN PIXEL SPACE OVER MILLIONS OF WEB SCREENSHOTS

Anonymous authors
Paper under double-blind review

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, improving accuracy by up to 18.1% over text-based 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). Finally, pixel representations enable an efficiency lever for RAG through image compression, effectively reducing reader input tokens by up to $3\times$ while maintaining accuracy. Our results question the necessity of text representations in web retrieval, suggesting that web RAG can operate in the web's native visual form while improving accuracy and reducing reader input tokens.

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 (Lewis et al., 2020; Guu et al., 2020; Karpukhin et al., 2020; Izacard & Grave, 2021; Nakano et al., 2021; Lyu et al., 2025a). 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 (Shao et al., 2024; Fang et al., 2025).

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 is complex, heavily engineered, and error-prone (Li et al., 2026a; Penedo et al., 2024; Tan et al., 2025). Even state-of-the-art parsers (Barbaresi, 2021; Wang et al., 2025a; Liu et al., 2025) 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 is hard to comprehend (Appendices D.2–D.3).

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 (Bai et al., 2025a; OpenAI, 2024; Gemini Team, 2025; Anthropic, 2024; Zhu et al., 2025; Wang et al., 2024b). 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

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

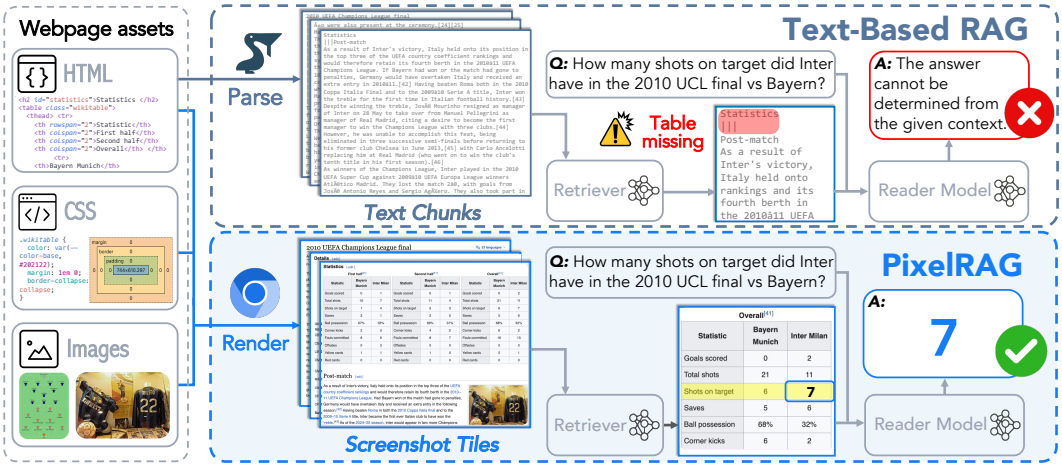


Figure 1: **Overview of PIXELRAG.** Text-based RAG (top) parses HTML into a text index and retrieves text chunks for the reader model. **PIXELRAG** (bottom, ours) renders each webpage, builds a visual index, and retrieves screenshot tiles for the reader — no HTML-to-text parser required.

such as PDFs (Faysse et al., 2025; Ma et al., 2024; Yu et al., 2025; Cho et al., 2024; Yan et al., 2026; Huo et al., 2026), 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 (Li et al., 2026b), which we further adapt to screenshot data via efficient contrastive fine-tuning with synthetic QA pairs and dynamic hard-negative mining (Xiong et al., 2021; Karpukhin et al., 2020). 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 (Kwiatkowski et al., 2019) and SimpleQA (Wei et al., 2024), where nearly all questions are answerable from text alone. PIXELRAG also delivers strong gains on multimodal QA, benchmarks over noisy news corpora (LiveVQA (Fu et al., 2025)), and agentic benchmarks (Jiang et al., 2024). 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 (Wei et al., 2025; Cheng et al., 2025). Training the VLM reader to tolerate lower resolutions further improves the accuracy–efficiency Pareto frontier, effectively reducing reader input tokens by up to 3× 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, effectively reducing reader input tokens by up to 3×. These gains suggest a shift toward pixel-space RAG as VLMs improve.

2 Related Work

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 (Karpukhin et al., 2020; Lewis et al., 2020; Guu et al., 2020), powering search-augmented agents (Nakano et al., 2021; Jin et al., 2025) and deep research systems (Tongyi DeepResearch Team et al., 2025). 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 (Li et al., 2024; Penedo et al., 2024): 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* (Barbaresi, 2021), *resiliparse* (Bevendorff et al., 2018), and *mwparserfromhell* (Kurtovic & contributors, 2026), or learned parsers such as ReaderLM (Wang et al., 2025a) and Dripper (Liu et al., 2025).

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 (Li et al., 2024; Penedo et al., 2024): a single extractor may discard over 40% of recoverable webpage text (Li et al., 2026a). Similar effects have also been observed in RAG systems, both in prior work (Fang et al., 2025) and in our own experiments (§5.2), where switching between the two parsers we evaluate end to end changes SimpleQA accuracy by 10.9%.

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 (Lu et al., 2024; Sun et al., 2026; Lyu et al., 2025b; Bai et al., 2025a). Concurrently, systems such as DeepSeek-OCR (Wei et al., 2025) and Glyph (Cheng et al., 2025) 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 (Faysse et al., 2025; Moreira et al., 2026; Yu et al., 2025), primarily on small-scale, visually rich benchmarks built from PDFs and slides (Macé et al., 2025; Zhu et al., 2024). 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. Closest to ours, DSE (Ma et al., 2024) retrieves Wikipedia screenshots, but only for a subset of NQ-related pages, capturing just the top 980×980 pixels of each page, and evaluates retrieval without an end-to-end reader.

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 and render success rates are in Appendices A.2 and D.8.

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 viewport are captured by scrolling.

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 ∼30M tiles for Wikipedia (7M articles) and ∼3.6M 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 (Faysse et al., 2025), and Nemotron ColEmbed (Moreira et al., 2026), 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 (Faysse et al., 2025; Shorten et al., 2026)), making multivector retrieval prohibitively expensive: each 875×1024-pixel tile emits ∼875 visual tokens[1] at 128 dimensions each (∼112K dimensions per tile), which would inflate the 30M-tile index to ∼6.7 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 (a measured ColPali comparison is in Appendix C.5). Here we use an off-the-shelf Qwen3-VL-Embedding-2B (Li et al., 2026b) and its fine-tuned variant (§4).

Index. At tens of millions of vectors, exact search is infeasible; we use a FAISS IVF index (Johnson et al., 2021) 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 ∼4× token reduction (Bai et al., 2025a). We study both trade-offs in §5.

[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 (Bai et al., 2025a).

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 and the training procedure in §4.1.

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, and no training data is generated from benchmark gold pages (Appendix C.6).

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 (Xiong et al., 2021) and treat all except p 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 7 in Appendix A.3.4 walks through a worked example. This pipeline uses a strong LLM to distill clean supervision into the embedding model, improving retrieval without human labels.

Contrastive training loss. We fine-tune the visual embedding model with InfoNCE loss (Oord et al., 2018) over mined triples and in-batch negatives. For a mini-batch of B triples $\{(q_i, p_i, \{n_{i,j}\}_{j=1}^{M})\}_{i=1}^{B}$,

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

with $\mathrm{sim}(\cdot,\cdot)$ 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 (Faysse et al., 2025) and the Swift framework (Zhao et al., 2024), applies LoRA only to the LLM backbone, as adapting the ViT degrades performance on document images. We observe the opposite: applying LoRA to both the LLM and the ViT consistently helps on webpage screenshots (Table 4), likely because near-duplicate tiles require finer discrimination of rendered text and table structure. Training takes under 3 hours on one H100.

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).

5.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate on three task families across two corpora. (1) *Text-centric Wikipedia QA* includes NQ (Kwiatkowski et al., 2019), NQ-Tables (Herzig et al., 2021), and SimpleQA (Wei

Table 1: Recall@3 and end-to-end QA accuracy on benchmarks over the Wikipedia corpus and the news corpus. –: not applicable (no retrieval; MMSearch lacks gold articles; *mwparserfromhell* is Wikipedia-specific).

	Wikipedia corpus										News corpus	
	NQ		NQ-Tables		SimpleQA		MMSearch		EVQA		LiveVQA	
Method	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
Text-based retrieval												
mwparserfromhell	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
Pixel-based retrieval												
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

et al., 2024), which are widely studied benchmarks whose questions are primarily answerable from textual content. (2) *Multimodal Wikipedia QA* includes MMSearch (Jiang et al., 2024)[2] and Encyclopedic VQA (Mensink et al., 2023), consisting of image-grounded queries. (3) *News VQA* includes LiveVQA (Fu et al., 2025), 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), whose pages we render without navigation bars and sidebars (Appendix A.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* (Barbaresi, 2021), a general-purpose extractor that retains tables and infoboxes as cell-delimited text, ranked best in both prior work (Penedo et al., 2024; Liu et al., 2025) and our own comparison against six alternatives (Appendix D.1); and *mwparserfromhell* (Kurtovic & contributors, 2026), a widely used Wikipedia-specific parser (Thrush et al., 2022; NeuML, 2024). Appendix C.3 adds learned extractors (ReaderLM-v2, Dripper; 44K-page subset due to cost) and BM25+dense hybrid retrieval; PIXELRAG leads all of them. Text indexes use 1024-token chunks, matching the $\sim$875 visual tokens per tile (§3.2) so both readers see a similar token budget, with the same embedding model (Li et al., 2026b) and FAISS (Johnson et al., 2021) index.

Configuration. Unless otherwise specified, the reader is Qwen3.5-4B (Qwen Team, 2026a) and receives k=3 retrieved items (text chunks or screenshot tiles). PIXELRAG (base) retrieves with the pretrained Qwen3-VL-Embedding-2B (Li et al., 2026b); 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 on the text-centric benchmarks, where fine-tuning adds up to 5.6% recall and 5.0% accuracy; paired tests confirm the gain over Trafilatura (e.g., NQ-Tables, McNemar p=0.0003; Appendix C.7). 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 15.5% over text retrieval 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

[2]MMSearch spans diverse web topics; we use the Wikipedia datastore as its retrieval corpus here.

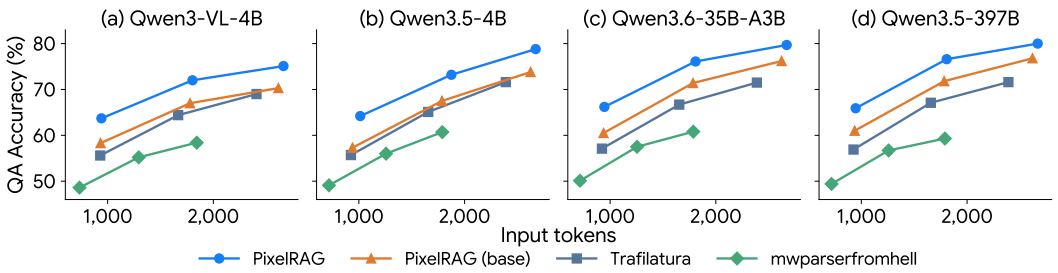


Figure 2: SimpleQA accuracy versus average input tokens across four reader models ($k \in \{1, 2, 3\}$).

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

		Accuracy (%)			Evidence Recall@3 (%)		
Type	n	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

Rows overlap: a question whose answer spans multiple content types (e.g., both a table and a paragraph) appears in several rows.

or Wikipedia-specific cleanup. Wikipedia fine-tuning transfers poorly to news; scaling training across domains remains future work.

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 ranks the gold article in its top 3 more often than Trafilatura (83.8% vs. 77.4%, Table 1), and the per-type breakdown shows where the gap comes from.

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 less susceptible 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.

Additionally, PIXELRAG overcomes three major failure modes of text-based retrieval (Appendix D.4): *parser loss* (42%), where no valid text evidence surfaces even in the top-100, often because linearization drops the answer at table or chunk boundaries (Fig. 10); *rank loss* (58%), where valid text evidence exists but ranks far outside top-3, e.g., displaced by keyword-rich infobox chunks as described above (Figs. 11–12); and *reader loss*, where linearized tables cause extraction failure even when evidence reaches the top-3 (Fig. 13).

Varying k and reader model. We vary $k \in \{1, 2, 3\}$ and the reader model on SimpleQA (Figure 2). 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 packs denser evidence than a 1024-token chunk (Wei et al., 2025). 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.

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

Table 3: Retrieval–reader modality ablation (SQA: SimpleQA; LVQA: LiveVQA).

Retrieval	Reader	SQA	LVQA
Screenshot	Screenshot	**73.8**	**70.3**
Screenshot	OCR text	72.2	64.5
Text	Rendered img	67.4	56.6
Text	Text	71.6	59.0
Text	HTML	59.8	56.6

Table 4: Embedding training recipe ablation. Ours uses dynamic hard negatives with an unfrozen ViT.

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**

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** (standard text-based RAG); and (5) **Text** → **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 → OCR outperforms Text → 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 → HTML), but this *underperforms* flat text on both benchmarks: HTML markup inflates context by 3.8×, 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 (Faysse et al., 2025), 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.

Matched training for text. A text retriever trained with the same contrastive pipeline still trails PIXELRAG by 3.0–8.8 R@1 on SimpleQA, NQ, NQ-Tables, and EVQA (Appendix C.4), so the gain comes from the pixel modality, not the training recipe.

5.4 AGENTIC SEARCH WITH PIXELRAG

We evaluate PIXELRAG as the search backend for a GPT-5 ReAct agent on MoNaCo (Wolfson et al., 2025), a multi-hop Wikipedia QA benchmark (setup in Appendix B.1). PIXELRAG attains the highest F1 at the lowest cost in our single-run comparison (Figure 3), ahead of text-chunk retrieval, Google (SerpApi, 2025), and DS-Serve (Liu et al., 2026) while costing 2–4× less. The controlled comparison is PIXELRAG versus text-chunk retrieval, which share the tool interface, prompt, and top-5 budget; Google and DS-Serve are external services included as reference points. We report token-level F1, the official MoNaCo metric, as answers are often multi-value lists where partial credit is appropriate. The agent also accumulates far fewer prompt tokens: 3.6M versus 37.5M for text retrieval (10× reduction; setup in Appendix B.1).

5.5 IMAGE COMPRESSION TO SAVE VISUAL TOKENS

Retrieved context makes up most of the reader's input, so shortening it directly lowers the reader's cost. Pixel inputs expose a compression knob unavailable to text: image resolution. Downsampling tiles via Lanczos resampling (Wikipedia contributors, 2025) reduces token count proportionally through VLM dynamic resolution (Bai et al., 2025a), 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×

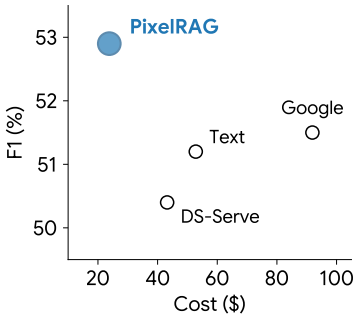


Figure 3: Agentic multi-hop QA on MoNaCo: F1 vs. total cost.

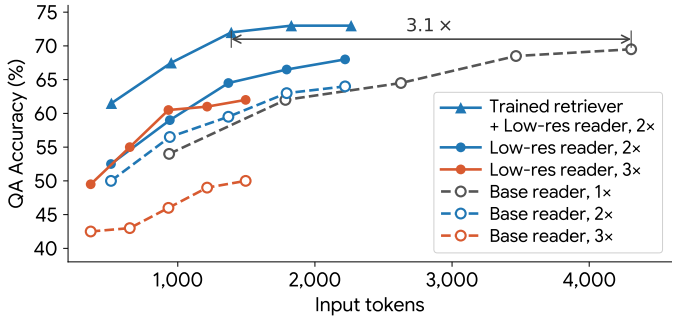
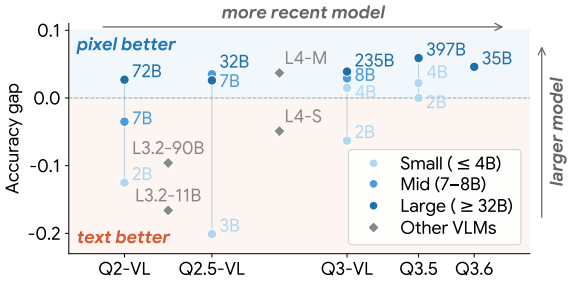
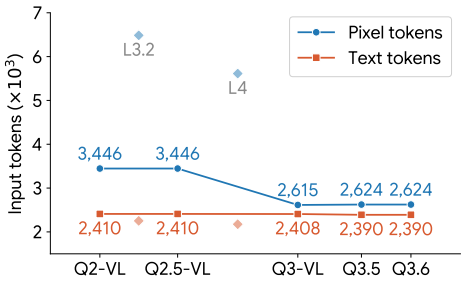


Figure 4: Accuracy vs. input tokens under image compression on a SimpleQA subset ($k \in \{1, \ldots, 5\}$). $c\times$ denotes the ratio.



(a) Accuracy gain over text retrieval



(b) Input token cost

Figure 5: VLM reading ability across model generations on SimpleQA. Q = Qwen; L3.2 = Llama-3.2-Vision; L4 = Llama-4. Markers denote parameter scale.

compression, the fine-tuned reader matches or exceeds native-resolution accuracy while halving tokens (Figure 4). 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 (Figure 5). Early VLMs struggle with screenshots: smaller models in the Llama-3.2-Vision and Qwen2-VL families trail text retrieval. In the Qwen3-VL family (Bai et al., 2025a), PixelRAG matches or beats text from the 4B model onward, and across newer readers the advantage reaches up to 5.9%, consistent with VLMs becoming increasingly capable at understanding text in pixels (Lyu et al., 2025b). Concurrently, newer VLMs encode the same image into fewer tokens (e.g., via larger vision patches), with Qwen3-VL reducing per-tile token cost by $\sim$24% over Qwen2-VL (Figure 5b; Table 9). 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.

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-to-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 effectively reduces reader input tokens by up to $3\times$, making PixelRAG increasingly practical with each generation.

Limitations and future directions are discussed in Appendix E.

AI USE STATEMENT

We used generative AI tools to assist implementation, visualization, and manuscript editing. LLMs that are part of the method (synthetic query generation, data filtering, and answer grading) are described in Section 4 and Appendices A.3 and B.2. The authors are responsible for the final methods, measurements, citations, and presentation.

ETHICS STATEMENT

Our datastores are built from a public Wikipedia snapshot and publicly accessible news articles; no human subjects or private data are involved. Screenshot retrieval preserves whatever a rendered page shows, including harmful or private content and, as for any RAG system over untrusted web pages, injected instructions (Greshake et al., 2023; Zou et al., 2025; Bagdasaryan et al., 2023), so deployment requires content filtering at rendering or indexing time.

REFERENCES

Anthropic. The Claude 3 model family: Opus, Sonnet, Haiku. Technical report, 2024. URL https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf. Technical Report.

Eugene Bagdasaryan, Tsung-Yin Hsieh, Ben Nassi, and Vitaly Shmatikov. Abusing images and sounds for indirect instruction injection in multi-modal LLMs. *arXiv preprint arXiv:2307.10490*, 2023.

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*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025b.

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 (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pp. 122–131, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-demo.15. URL https://aclanthology.org/2021.acl-demo.15/.

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 (eds.), *Advances in Information Retrieval. 40th European Conference on IR Research (ECIR 2018)*, Lecture Notes in Computer Science, Berlin Heidelberg New York, March 2018. Springer.

Jiale Cheng, Yusen Liu, Xinyu Zhang, Yulin Fei, Wenyi Hong, Ruiliang Lyu, Weihan Wang, Zhe Su, Xiaotao Gu, Xiao Liu, et al. Glyph: Scaling context windows via visual-text compression. *arXiv preprint arXiv:2510.17800*, 2025.

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

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.

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.

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.

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.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *ACM Workshop on Artificial Intelligence and Security (AISec)*, 2023.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. REALM: Retrieval-augmented language model pre-training. In *International Conference on Machine Learning*, pp. 3929–3938, 2020.

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*, pp. 512–519, 2021.

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.

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*, pp. 874–880, 2021.

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.

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.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2021. doi: 10.1109/TBDATA.2019.2921572. URL https://doi.org/10.1109/TBDATA.2019.2921572.

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*, pp. 6769–6781, 2020.

Kiwix Association. Kiwix — offline reader for web content. https://kiwix.org, 2007. Open-source offline browser using the ZIM archive format.

Ben Kurtovic and contributors. mwparserfromhell: A python parser for mediawiki wikicode. https://github.com/earwig/mwparserfromhell, 2026.

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.

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.

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.

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*, pp. 5836–5861, 2026a.

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

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2023.

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)*, pp. 41631–41633, 2026. doi: 10.1609/aaai.v40i48.42363.

Mengjie Liu, Jiahui Peng, Wenchang Ning, et al. Dripper: Token-efficient main HTML extraction with a lightweight LM. *arXiv preprint arXiv:2511.23119*, 2025.

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.

Xinxi Lyu, Michael Duan, Rulin Shao, Pang Wei Koh, and Sewon Min. Frustratingly simple retrieval improves challenging, reasoning-intensive benchmarks. *arXiv preprint arXiv:2507.01297*, 2025a.

Zhiheng Lyu, Xueguang Ma, and Wenhu Chen. Pixelworld: How far are we from perceiving everything as pixels? *arXiv preprint arXiv:2501.19339*, 2025b.

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

Quentin Macé, António Loison, and Manuel Faysse. Vidore benchmark v2: Raising the bar for visual retrieval. *arXiv preprint arXiv:2505.17166*, 2025.

Thomas Mensink, Jasper Uijlings, Lluis 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*, pp. 3082–3092, 2023.

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.

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.

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.

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.

NeuML. Neuml/wikipedia: Wikipedia text dataset. https://huggingface.co/datasets/NeuML/wikipedia, 2024. Text extracted from Wikipedia XML dumps via mwparserfromhell.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

OpenAI. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

OpenAI. Introducing GPT-4.1 in the API, April 2025. URL https://openai.com/index/gpt-4-1/. Blog post.

OpenAI. Images and vision. https://developers.openai.com/api/docs/guides/images-vision, 2026. OpenAI API documentation.

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.

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

Qwen Team. Qwen3.6: Towards real world agents. https://qwen.ai/blog?id=qwen3.6/, 2026b.

SerpApi. SerpApi: Google search api. https://serpapi.com, 2025.

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.

Connor Shorten, Augustas Skaburskas, Daniel M Jones, Charles Pierse, Roberto Esposito, John Trengrove, Etienne Dilocker, and Bob van Luijt. Irpapers: A visual document benchmark for scientific retrieval and question answering. *arXiv preprint arXiv:2602.17687*, 2026.

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.

Jiejun Tan, Zhicheng Dou, Wen Wang, Mang Wang, Weipeng Chen, and Ji-Rong Wen. HtmlRAG: HTML is better than plain text for modeling retrieved knowledge in RAG systems. In *Proceedings of the ACM Web Conference 2025 (WWW)*, 2025.

Tristan Thrush, Helen Ngo, Nathan Lambert, and Douwe Kiela. Online language modelling data pipeline. https://github.com/huggingface/olm-datasets, 2022.

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.

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*, 2025a.

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*, 2024a.

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*, 2025b.

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, 2024b.

Haoran Wei, Yaofeng Sun, and Yukun Li. Deepseek-ocr: Contexts optical compression. *arXiv preprint arXiv:2510.18234*, 2025.

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.

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.

Wikipedia contributors. Lanczos resampling. https://en.wikipedia.org/wiki/Lanczos_resampling, 2025. Accessed: 2026-04-29.

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.

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.

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.

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.

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. URL https://arxiv.org/abs/2408.05517.

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.

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.

Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. PoisonedRAG: Knowledge corruption attacks to retrieval-augmented generation of large language models. In *USENIX Security*, 2025.

TECHNICAL APPENDICES AND SUPPLEMENTARY MATERIAL

Contents

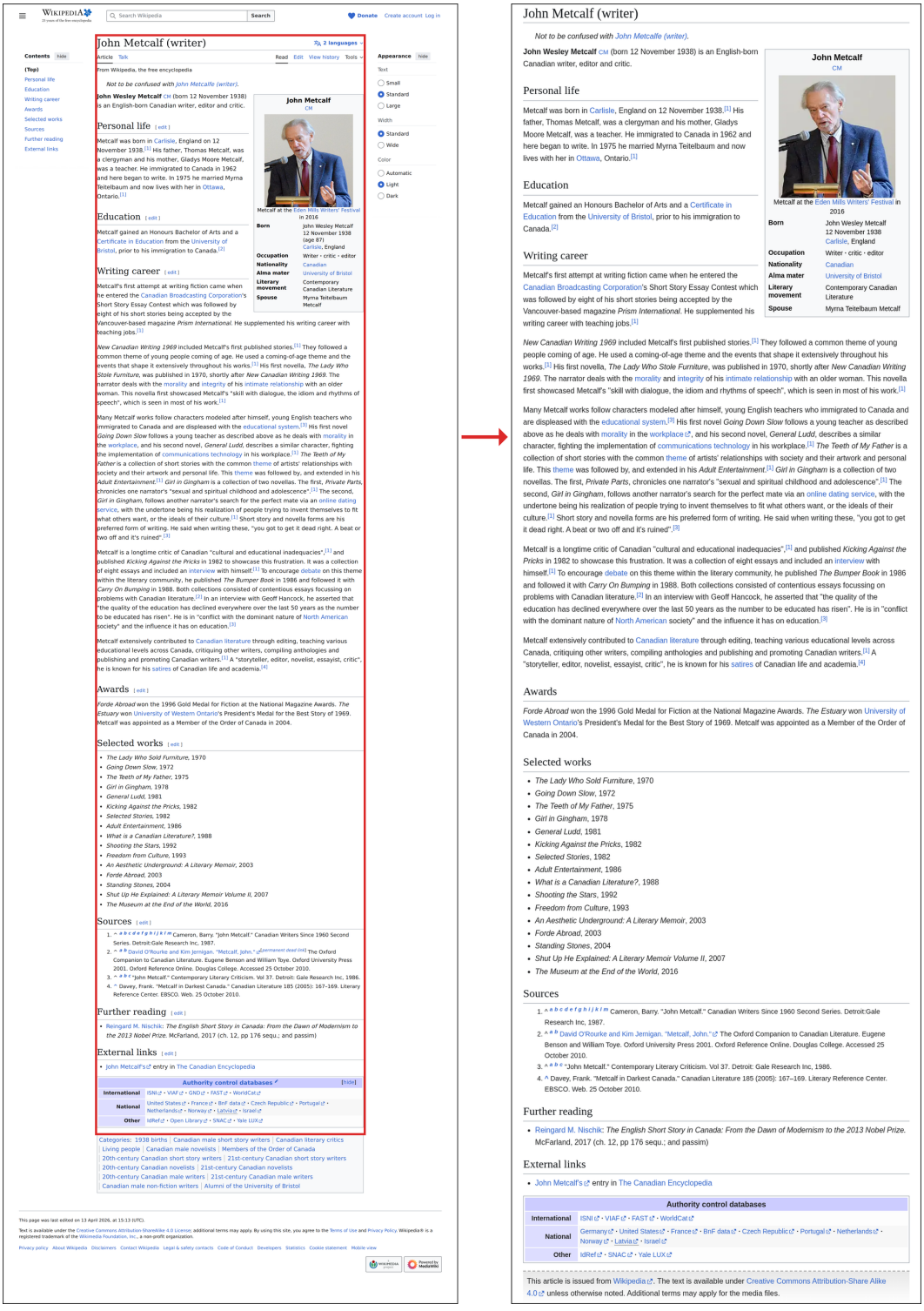
A System & Implementation Details
A.1 Rendering Pipeline 15
A.2 Datastore Fetching 17
A.3 Embedding Training: Data Recipe Details and Prompts 17
A.4 Mini-Datastore for Embedding Ablations 18
A.5 Reader Fine-tuning: Method and Results 19
B Evaluation Protocol
B.1 Benchmark Details 19
B.2 Grading Protocol 20
C Additional Results
C.1 Results with Alternative Readers 20
C.2 Full Results of Scaling VLM Performance 20
C.3 Stronger Text Baselines 21
C.4 Matched Training for the Text Retriever 23
C.5 Multi-Vector Visual Retrieval 23
C.6 Training Data and Evaluation Benchmarks 23
C.7 Statistical Significance 24
C.8 End-to-End Latency and Storage 24
D Analysis
D.1 Wikipedia Text Extractor Comparison 24
D.2 Visual Information Loss During HTML Parsing 24
D.3 Retrieval Signal Loss Under Text Linearization 25
D.4 Detailed Failure Decomposition 26
D.5 HTML DOM Lookup Baseline: Setup and Analysis 31
D.6 Directly RAG on Raw HTML Data 32
D.7 Tables Split Across Tiles 33
D.8 Rendering Success on the News Corpus 33
E Limitations and Future Work 34
F Prompt Listings 34

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 ($\sim$5.6 TB for Wikipedia and $\sim$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 (Wang et al., 2025b), reducing the embedding storage cost as well. Query-time costs, including re-rendering, are in Appendix C.8.



John Metcalf (writer)

Not to be confused with John Metcalfe (writer).

John Wesley Metcalf CM (born 12 November 1938) is an English-born Canadian writer, editor and critic.

John Metcalf
CM

Metcalf at the Eden Mills Writers' Festival in 2016

Born	John Wesley Metcalf 12 November 1938 (age 87) Carlisle, England
Occupation	Writer · critic · editor
Nationality	Canadian
Alma mater	University of Bristol
Literary movement	Contemporary Canadian Literature
Spouse	Myrna Teitelbaum Metcalf

Personal life

Metcalf was born in Carlisle, England on 12 November 1938.[1] His father, Thomas Metcalf, was a clergyman and his mother, Gladys Moore Metcalf, was a teacher. He immigrated to Canada in 1962 and here began to write. In 1975 he married Myrna Teitelbaum and now lives with her in Ottawa, Ontario.[1]

Education

Metcalf gained an Honours Bachelor of Arts and a Certificate in Education from the University of Bristol, prior to his immigration to Canada.[2]

Writing career

Metcalf's first attempt at writing fiction came when he entered the Canadian Broadcasting Corporation's Short Story Essay Contest which was followed by eight of his short stories being accepted by the Vancouver-based magazine *Prism International*. He supplemented his writing career with teaching jobs.[1]

New Canadian Writing 1969 included Metcalf's first published stories.[1] They followed a common theme of young people coming of age. He used a coming-of-age theme and the events that shape it extensively throughout his works.[1] His first novella, *The Lady Who Stole Furniture*, was published in 1970, shortly after *New Canadian Writing 1969*. The narrator deals with the morality and integrity of his intimate relationship with an older woman. This novella first showcased Metcalf's "skill with dialogue, the idiom and rhythms of speech", which is seen in most of his work.[1]

Many Metcalf works follow characters modeled after himself, young English teachers who immigrated to Canada and are displeased with the educational system.[3] His first novel *Going Down Slow* follows a young teacher as described above as he deals with morality in the workplace, and his second novel, *General Ludd*, describes a similar character, fighting the implementation of communications technology in his workplace.[1] *The Teeth of My Father* is a collection of short stories with the common theme of artists' relationships with society and their artwork and personal life. This theme was followed by, and extended in his *Adult Entertainment*.[1] *Girl in Gingham* is a collection of two novellas. The first, *Private Parts*, chronicles one narrator's "sexual and spiritual childhood and adolescence".[1] The second, *Girl in Gingham*, follows another narrator's search for the perfect mate via an online dating service, with the undertone being his realization of people trying to invent themselves to fit what others want, or the ideals of their culture.[1] Short story and novella forms are his preferred form of writing. He said when writing these, "you got to get it dead right. A beat or two off and it's ruined".[3]

Metcalf is a longtime critic of Canadian "cultural and educational inadequacies",[1] and published *Kicking Against the Pricks* in 1982 to showcase this frustration. It was a collection of eight essays and included an interview with himself.[1] To encourage debate on this theme within the literary community, he published *The Bumper Book* in 1986 and followed it with *Carry On Bumping* in 1988. Both collections consisted of contentious essays focussing on problems with Canadian literature.[2] In an interview with Geoff Hancock, he asserted that "the quality of the education has declined everywhere over the last 50 years as the number to be educated has risen". He is in "conflict with the dominant nature of North American society" and the influence it has on education.[3]

Metcalf extensively contributed to Canadian literature through editing, teaching various educational levels across Canada, critiquing other writers, compiling anthologies and publishing and promoting Canadian writers.[1] A "storyteller, editor, novelist, essayist, critic", he is known for his satires of Canadian life and academia.[4]

Awards

Forde Abroad won the 1996 Gold Medal for Fiction at the National Magazine Awards. *The Estuary* won University of Western Ontario's President's Medal for the Best Story of 1969. Metcalf was appointed as a Member of the Order of Canada in 2004.

Selected works

- *The Lady Who Sold Furniture*, 1970
- *Going Down Slow*, 1972
- *The Teeth of My Father*, 1975
- *Girl in Gingham*, 1978
- *General Ludd*, 1981
- *Kicking Against the Pricks*, 1982
- *Selected Stories*, 1982
- *Adult Entertainment*, 1986
- *What is a Canadian Literature?*, 1988
- *Shooting the Stars*, 1992
- *Freedom from Culture*, 1993
- *An Aesthetic Underground: A Literary Memoir*, 2003
- *Forde Abroad*, 2003
- *Standing Stones*, 2004
- *Shut Up He Explained: A Literary Memoir Volume II*, 2007
- *The Museum at the End of the World*, 2016

Sources

1. ^ a b c d e f g h i j k l m Cameron, Barry. "John Metcalf." Canadian Writers Since 1960 Second Series. Detroit:Gale Research Inc, 1987.
2. ^ a b David O'Rourke and Kim Jernigan. "Metcalf, John." The Oxford Companion to Canadian Literature. Eugene Benson and William Toye. Oxford University Press 2001. Oxford Reference Online. Douglas College. Accessed 25 October 2010.
3. ^ a b c "John Metcalf." Contemporary Literary Criticism. Vol 37. Detroit: Gale Research Inc, 1986.
4. ^ Davey, Frank. "Metcalf in Darkest Canada." Canadian Literature 185 (2005): 167–169. Literary Reference Center. EBSCO. Web. 25 October 2010.

Further reading

- Reingard M. Nischik: *The English Short Story in Canada: From the Dawn of Modernism to the 2013 Nobel Prize*. McFarland, 2017 (ch. 12, pp 176 sequ.; and passim)

External links

- John Metcalf's entry in The Canadian Encyclopedia

Authority control databases	
International	ISNI · VIAF · FAST · WorldCat
National	Germany · United States · France · BnF data · Czech Republic · Portugal · Netherlands · Norway · Latvia · Israel
Other	IdRef · SNAC · Yale LUX

This article is issued from Wikipedia. The text is available under Creative Commons Attribution-Share Alike 4.0 unless otherwise noted. Additional terms may apply for the media files.

Figure 6: A Wikipedia page as it appears online in a browser (left) and after our rendering pipeline (right). We strip surrounding UI elements and whitespace, keeping only the article content, which is then tiled for indexing.

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 $\sim$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 blocklist. 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 $\rightarrow$ 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 Figure 17 (Appendix F). Figure 7 walks through a worked example.

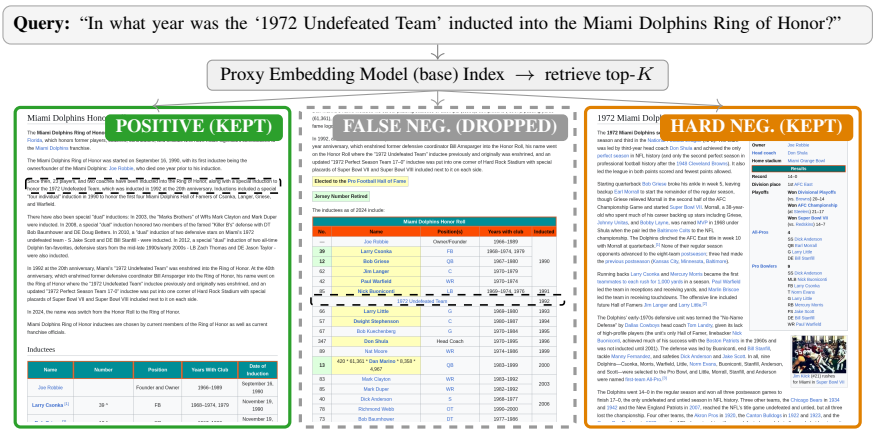


Figure 7: **Hard-negative mining with false-negative filtering.** The LLM correctly answers the query from both the positive tile (left, kept) and a second tile (center, dropped as false negative), but cannot answer from a topically adjacent page (right, kept as hard negative).

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 (Figure 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. 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. The production run that generated and filtered ∼195K candidates into these ∼40K pairs cost about $200 in API usage; including earlier pipeline ablations and development runs, total API spend was about $1,000. Generation and filtering use different models (`gemini-3.1-flash-lite-preview` generates, GPT-4o filters), and GPT-4.1, our evaluation judge, is not involved in producing training data. A manual spot check of randomly sampled generated queries and hard-negative decisions found no systematic errors.

A.4 Mini-Datastore for Embedding Ablations

Evaluating every embedding checkpoint against the full 30M-tile Wikipedia datastore is prohibitively expensive. We instead construct a compact mini-datastore for rapid iteration. We sample 400 SimpleQA queries; they are disjoint from the 1,000 SimpleQA test questions in Table 1. For each query, the mini-datastore 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-datastore approximates 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-datastore. We will publicly release this mini-datastore 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.3.5), giving us SFT triples (q, p, a) at no additional annotation cost. To simulate the retrieval setting, we use the embedding model from §4 to retrieve $k \in \{1, \ldots, 6\}$ tiles for q over the datastore, and add p to the set if it is not already present. All tiles are downsampled via Lanczos resampling (Wikipedia contributors, 2025) 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: $\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(q,\mathcal{T},a)} \sum_t \log P_\phi(a_t \mid \mathcal{T}, q, a_{<t})$, where $\mathcal{T}$ is the retrieved tile set.

Evaluation. Table 5 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, $+9.3$pp above the compression-only baseline at the same c and $+4.2$pp above the uncompressed ceiling itself; SFT at $3\times$ reaches 0.910, $+17.2$pp above its compression-only baseline and matching the uncompressed ceiling on average ($+0.5$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 1.8pp of its $k=3$ peak across the full k sweep, consistent with the "ignore distractor / focus on gold tile" training goal described above.

Table 5: 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 @ $1\times$ (no SFT, ceiling)	0.958	0.912	0.892	0.856	0.905
Base @ $2\times$ (compression only, no SFT)	0.908	0.862	0.852	0.794	0.854
SFT @ $2\times$	**0.946**	**0.950**	**0.954**	**0.936**	**0.947**
Base @ $3\times$ (compression only, no SFT)	0.830	0.726	0.710	0.684	0.738
SFT @ $3\times$	0.904	0.918	0.932	0.884	0.910

B Evaluation Protocol

B.1 Benchmark Details

Table 6 summarizes the evaluation benchmarks used in this paper.

Encyclopedic VQA subset. We evaluate on the *landmarks* subset (Google Landmarks v2 (Weyand et al., 2020)) 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.

Table 6: Benchmark details. All Wikipedia benchmarks query the same 30M-tile 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	(Kwiatkowski et al., 2019)
NQ-Tables	Text-centric Wiki QA	1,000	GPT-4.1 judge	(Herzig et al., 2021)
SimpleQA	Text-centric Wiki QA	1,000	GPT-4.1 judge	(Wei et al., 2024)
MMSearch	Multimodal Wiki QA	300	GPT-4.1 judge	(Jiang et al., 2024)
Encyclopedic VQA	Multimodal Wiki QA	749	GPT-4.1 judge	(Mensink et al., 2023)
LiveVQA-2025	News VQA	6,632*	Accuracy	(Fu et al., 2025)

$\star$ CNN/BBC/AP News subset; the full LiveVQA dataset has 26,888 QA pairs.

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).

MoNaCo setup. We evaluate on the full MoNaCo benchmark (Wolfson et al., 2025) (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 (SerpApi, 2025), and DS-Serve (Liu et al., 2026) (open-source neural retrieval endpoint). The agent issues up to 20 search queries per question with top-k=5 results per query. Retrieved tiles are passed to GPT-5 with OpenAI's default `auto` image-detail setting, under which the API chooses the input resolution and downsamples most tiles (OpenAI, 2026).

B.2 GRADING PROTOCOL

All Wikipedia QA benchmarks (NQ, NQ-Tables, SimpleQA, MMSearch, Encyclopedic VQA) use LLM-as-judge grading with GPT-4.1 (OpenAI, 2025) at temperature 0, seed 42, and `max_tokens=1000`. Following Wei et al. (2024), 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). The judged benchmarks consist of short factoid questions with constrained answers, the setting where LLM judges agree closely with human graders. As a stability check, two independent judges (GPT-4.1 and GPT-4o) agree on 95.1% of a 102-example audit (Cohen's κ=0.90). Retrieval recall uses exact gold-URL matching, and LiveVQA and MoNaCo use their deterministic official metrics, so these results do not depend on an LLM judge. 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. Table 7 repeats the main results at top-k=1 with the Qwen3-VL-4B reader. All directional findings are identical across reader variants. The tuned retriever (LoRA) improves over the base checkpoint on most benchmarks: on SimpleQA with the VL-4B reader at top-1, accuracy rises from 58.3% to 63.7%.

C.2 FULL RESULTS OF SCALING VLM PERFORMANCE

Table 9 reports the complete 29-model sweep summarized in Figure 5. We evaluate 26 non-reasoning and 3 reasoning-mode VLMs on SimpleQA under pixel retrieval (top-k=3, PIXELRAG base) and

Table 7: End-to-end results at top-k=1 with Qwen3-VL-4B reader (cf. Table 1).

	Wikipedia corpus									News corpus	
	NQ		NQ-Tables		SimpleQA		MMSearch	EVQA		LiveVQA	
Method	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
Text-based retrieval											
mwparserfromhell	28.1	23.1	17.8	14.9	58.8	48.6	18.1	1.8	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
Pixel-based retrieval											
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 8: 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

text retrieval (Trafilatura, same k), with all other variables held fixed. The crossover point — where pixel retrieval first matches text — occurs at Qwen3-VL-4B (70.5% pixel vs. 69.0% text). Below this capability threshold (e.g., Qwen2-VL-2B, LLaVA-1.5), models cannot reliably read text rendered in pixels, and text retrieval dominates. Above it, most models tested favor pixel retrieval, by up to +5.9 pp. Reasoning-mode models (bottom rows) 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). Concurrently, successive VLM generations encode the same tile into fewer input tokens: Qwen2-VL averages 3,446 tokens per top-3 pixel query, Qwen3-VL reduces this to 2,615 (-24%), demonstrating that the cost of pixel retrieval decreases with each model generation without any pipeline change.

C.3 STRONGER TEXT BASELINES

Our full-corpus text baseline uses Trafilatura, the top-ranked of seven extractors (Appendix D.1), and Appendices D.5 and D.6 evaluate pipelines that keep the raw HTML with its DOM, table, and infobox markup. Here we add learned extractors and sparse-dense hybrid retrieval.

Shared page pool. Re-extracting all 7M articles once per extractor is prohibitively expensive (Table 10), so the extractor comparison runs on one shared pool of 44,528 pages: every benchmark gold article, the ablation-store articles, and distractors contributed equally by prior retrieval runs of both modalities. Each method retrieves over its own extraction of these pages (200K–700K chunks), with the same chunking, embedding model, reader, and judge.

Structure-preserving and learned extractors. Table 10 compares Trafilatura with the learned extractors Dripper (Liu et al., 2025) and ReaderLM-v2 (Wang et al., 2025a). Dripper improves on Trafilatura only marginally at roughly 700× the extraction cost, since it re-labels DOM blocks as content or boilerplate, which Trafilatura's heuristics already handle on Wikipedia's uniform template; ReaderLM-v2 trails every pipeline, because it regenerates each page token by token and, on long table-dense articles, truncates and paraphrases instead of transcribing (202K chunks against Trafilatura's 270K on the same pages). PIXELRAG leads every extractor on every metric, and PIXELRAG (base), with no fine-tuning, leads all of them on R@1.

Sparse-dense hybrid retrieval. We run hybrid retrieval at full corpus scale: BM25 over all 15.7M Trafilatura chunks, fused with the dense text index by reciprocal rank fusion (Table 11). Fusion is the strongest text configuration, and the pixel index leads it on every benchmark. Sparse, dense, and

Table 9: Full reader-model sweep on SimpleQA under pixel vs. text retrieval (top-k=3). Token usage is average input and output tokens per query. This is the complete version of Figure 5.

	Pixel retrieval			Text retrieval		
Reader model	Acc	in tok	out tok	Acc	in tok	out tok
LLaVA-1.5-7B (Liu et al., 2023)‡	0.092	1817	32	0.504	2537	33
Llama-3.2-11B-Vision (Meta AI, 2024)†	0.534	6485	30	0.700	2252	26
Llama-3.2-90B-Vision (Meta AI, 2024)†	0.574	6485	32	0.670	2252	39
Llama-4-Scout-17B-16E (Meta AI, 2026)	0.618	5613	93	0.667	2175	60
Llama-4-Maverick-17B-128E (Meta AI, 2026)	0.756	5613	57	0.719	2175	73
Qwen2-VL-2B (Wang et al., 2024a)	0.439	3446	27	0.564	2410	41
Qwen2-VL-7B (Wang et al., 2024a)	0.628	3446	15	0.663	2410	26
Qwen2-VL-72B (Wang et al., 2024a)	0.745	3446	23	0.718	2410	14
Qwen2.5-VL-3B (Bai et al., 2025b)	0.460	3446	20	0.661	2410	17
Qwen2.5-VL-7B (Bai et al., 2025b)	0.713	3446	16	0.678	2410	21
Qwen2.5-VL-32B (Bai et al., 2025b)	0.717	3446	29	0.691	2410	56
Qwen2.5-VL-72B (Bai et al., 2025b)	0.745	3446	17	0.694	2346	26
Qwen3-VL-2B (Bai et al., 2025a)	0.613	2615	229	0.676	2408	34
Qwen3-VL-4B (Bai et al., 2025a)	0.705	2615	70	0.690	2408	62
Qwen3-VL-8B (Bai et al., 2025a)	0.726	2615	95	0.697	2408	100
Qwen3-VL-30B-A3B (Bai et al., 2025a)	0.733	2615	109	0.715	2344	37
Qwen3-VL-32B (Bai et al., 2025a)	0.742	2615	52	0.710	2344	43
Qwen3-VL-235B-A22B (Bai et al., 2025a)	0.759	2615	52	0.720	2408	42
Qwen3.5-0.8B (Qwen Team, 2026a)	0.602	2624	73	0.593	2390	50
Qwen3.5-2B (Qwen Team, 2026a)	0.667	2624	116	0.667	2390	104
Qwen3.5-4B (Qwen Team, 2026a)	0.738	2624	118	0.716	2459	95
Qwen3.5-9B (Qwen Team, 2026a)	0.750	2624	115	0.701	2390	83
Qwen3.5-27B (Qwen Team, 2026a)	0.759	2624	88	0.711	2390	69
Qwen3.5-35B-A3B (Qwen Team, 2026a)	0.737	2622	148	0.713	2390	89
Qwen3.6-27B (Qwen Team, 2026b)	0.754	2624	73	0.715	2390	65
Qwen3.6-35B-A3B (Qwen Team, 2026b)	0.751	2624	110	0.705	2390	50
Qwen3.5-4B (reasoning) (Qwen Team, 2026a)	0.745	2622	1311	0.697	2457	1597
Qwen3.5-397B-A17B (reasoning) (Qwen Team, 2026a)	0.769	2622	1081	0.710	2461	1188
Qwen3.6-35B-A3B (reasoning) (Qwen Team, 2026b)	0.763	2622	1254	0.717	2457	1005

† Llama-3.2-Vision models are hard-capped at one image per prompt by their cross-attention adapter architecture; the pixel column uses k=1.
‡ LLaVA-1.5-7B (2023-10) uses Vicuna-7B + CLIP ViT-L projection adapter — included as a historical baseline.

Table 10: Structured-text extraction baselines on a shared 44,528-page pool, with identical chunking, embedding model, reader, and judge. R@k is article-level; QA is end-to-end accuracy; SQA = SimpleQA, NQT = NQ-Tables. Extraction time is estimated for all 7.13M articles.

	R@1			R@3			QA			
Method	SQA	NQ	NQT	SQA	NQ	NQT	SQA	NQ	NQT	Extraction time
Trafilatura	67.7	31.5	25.9	83.5	57.3	54.7	71.4	56.9	40.6	3.5 h, one 128-core machine
Dripper (MinerU-HTML)	68.7	32.1	29.5	84.0	60.8	56.3	71.5	57.7	41.9	2,370 h, one H200
ReaderLM-v2	64.5	26.9	23.3	80.3	52.6	47.5	61.4	49.3	33.8	4,110 h, one H200
PIXELRAG (base)	70.9	40.8	35.1	85.5	63.4	57.6	72.0	56.9	44.1	no text extraction
PIXELRAG	**76.3**	**45.5**	**42.3**	**88.6**	**69.9**	**65.3**	**74.9**	**58.4**	**47.4**	no text extraction

fused retrieval sit on a different axis from whether the index holds text or pixels; reranking is likewise a second-stage re-scorer that applies to either modality, and we report first-stage retrieval throughout.

Table 11: Sparse, dense, and hybrid text retrieval over the full 7M-article corpus (15.7M Trafilatura chunks) against PIXELRAG. Article-level R@1; hybrid fuses BM25 and dense rankings with RRF (k=60). The PIXELRAG row comes from the production-index run; the pool results in Tables 10 and 12 re-embed the queries, so the two differ by under one point.

Retrieval	SimpleQA	NQ	NQ-Tables
BM25 over text	67.1	27.5	21.9
Dense over text	67.1	37.8	24.1
BM25 + dense over text (RRF)	73.2	41.1	38.3
Dense over pixels (PIXELRAG)	**76.3**	**46.4**	**42.4**

C.4 MATCHED TRAINING FOR THE TEXT RETRIEVER

Table 12: Matched contrastive training for the text retriever (article-level R@1, full 7M-article corpus).

Retriever	SQA	NQ	NQT	EVQA
Text, matched-trained	73.3	39.0	33.6	14.2
PIXELRAG	**76.3**	**46.4**	**42.4**	**17.8**

PIXELRAG (base) uses no synthetic training and already outperforms the text baselines in Table 1, so the pixel advantage does not depend on fine-tuning. To compare the two modalities under equal training, we apply the pipeline of §4 to the text embedding model: each text chunk is a positive, the same query generator produces a query grounded in it, the base model mines hard negatives, the same false-negative filter keeps valid ones, and the same model size, InfoNCE objective, and hyperparameters are used. PIXELRAG stays ahead of the matched-trained text retriever on all four benchmarks (Table 12).

C.5 MULTI-VECTOR VISUAL RETRIEVAL

We compare against ColPali v1.3 (Faysse et al., 2025) on the ablation mini-datastore (Appendix A.4), with identical candidate pools and queries (Table 13). Late interaction stores one vector per image patch rather than one per tile, and the resulting index size rules out late interaction at the scale this paper targets, where single-vector retrieval is the practical choice; PIXELRAG matches ColPali at R@10.

Table 13: Single-vector PIXELRAG versus late-interaction ColPali on the ablation mini-datastore (400 queries; article-level recall, identical candidate pools and queries). Index sizes are measured per tile and extrapolated to the full Wikipedia datastore.

Retriever	R@1	R@3	R@10	Vectors per tile	Full index (fp16)
ColPali v1.3 (late interaction)	80.0	93.3	98.8	1,031 × 128-d	7.9 TB
PIXELRAG (base)	68.5	84.3	96.5	1 × 2,048-d	120 GB
PIXELRAG	74.8	88.8	98.8	1 × 2,048-d	120 GB

C.6 TRAINING DATA AND EVALUATION BENCHMARKS

The retriever's training data is synthesized entirely from the datastore and never uses benchmark queries, labels, or gold pages. A normalized 13-gram audit confirms that no synthetic training query overlaps with any evaluation query. Independently, PIXELRAG (base) uses no synthetic training data and still outperforms the text baselines.

C.7 Statistical Significance

All methods are evaluated on the same examples, so we test the paired difference between PixelRAG and the Trafilatura baseline with a paired bootstrap for 95% confidence intervals and McNemar's test. On NQ-Tables, the accuracy gain of $+6.3$ has a 95% CI of $[+2.9, +9.5]$ (p=0.0003).

C.8 End-to-End Latency and Storage

Section 5.5 reports savings in reader input tokens only; here we report end-to-end latency and storage, measured on one host (B300 GPU, our serving stack, p50 over 200 SimpleQA queries). Query embedding takes 22 ms in both pipelines. Once tokens reach the reader, image and text tokens prefill at the same rate (81 ms time to first token for three tiles, ∼2,679 tokens, against 70 ms for three chunks, ∼2,200 tokens) and decode at the same speed (∼331 vs. ∼332 tokens/s). Retrieval, not the reader, dominates both pipelines: IVF search takes 1.13 s over the pixel index against 0.63 s over the 15.7M-chunk text index, for an end-to-end p50 of ∼1.9 s against ∼1.4 s. The difference comes from vector count; both indexes use the same dimension and precision. Only the embedding index is needed at serving time (120 GB against 64 GB in fp16). The rendered tile store (5.6 TB against 52 GB of extracted passages) is an optional cache: render-on-demand (Appendix E) keeps no tiles and re-renders retrieved pages in 1–3 s each. Offline, rendering and indexing all 7M articles takes ∼2 days on one machine and retriever fine-tuning under 3 hours on one H100; the text pipeline instead pays its extraction cost on every corpus refresh (Table 10).

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, cleanliness, tabular data, hierarchy, and overall quality, evaluated over 50 diverse Wikipedia pages (Table 14). `trafilatura` 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.

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 14: 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	`trafilatura`	**6.40**	6.56	7.46	6.24	**4.38**	**8.04**	**36**
2	`resiliparse`	6.34	**6.64**	**7.60**	6.02	4.02	7.16	31
3	BeautifulSoup	6.06	6.26	7.20	6.88	3.86	5.88	29
4	`mwparserfromhell`	5.56	5.60	6.52	6.24	3.16	6.60	15
5	`wikiextractor`	5.20	5.26	6.10	**7.54**	2.82	5.26	8
6	raw wikitext	4.60	4.92	6.12	3.62	2.76	5.86	3
7	`jina_reader`	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.




```
[edit] Argentina | France | | Man of the Match: Assistant referees:
```

Formation diagrams, lineup tables, penalty icons: all lost.



Monthly awards [edit]

Month	Manager of the Month		Player of the Month		Goal of the Month	
	Manager	Club	Player	Club	Player	Club
August	Nuno Espírito Santo	Tottenham Hotspur	Michail Antonio	West Ham United	Danny Ings	Aston Villa
September	Mikel Arteta	Arsenal	Cristiano Ronaldo	Manchester United	Andros Townsend	Everton
October	Thomas Tuchel	Chelsea	Mohamed Salah	Liverpool	Mohamed Salah	Liverpool
November	Pep Guardiola	Manchester City	Trent Alexander-Arnold	Liverpool	Rodri	Manchester City
December			Raheem Sterling	Manchester City	Alexandre Lacazette	Arsenal

```
Monthly awards Month Manager of the Month Player of the Month Goal of the Month ... August Nuno Espírito Santo Tottenham Hotspur ... December Raheem Sterling Manchester City Alexandre Lacazette Arsenal
```

Merged cells flattened; December's Manager of the Month lost.

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.

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 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 1.

Text ranking (1024-token chunks)

rank 1, score 0.715

```
Indigenous peoples in Ecuador;
Paraguay; Costa Rica; Argentina;
Peru; Canada; Colombia; Brazil;
...
```

rank 2, score 0.647

```
Portal:  Indigenous peoples of
the Americas / box-footer
```

rank 3, score 0.639

```
| Honduras | 49% | 19% | 10% |
... | Mexico | 58% | 9% | 21% |
...
```

⋮

rank 38, score 0.580

```
... INEGI, Mexico ... United
States Census Bureau ...
```

Pixel ranking (875×1024 tile, 880 tokens)

rank 2, score 0.616



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.

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 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, `<ul>/<ol>` → 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 15 pairs each question's text and screenshot outcomes. A question is classified as FM1 (parser loss) when PixelRAG retrieval surfaces valid evidence in top-3 but no valid text chunk appears anywhere in the retrieved top-100. We use this as an operational proxy for parsing loss: in 95.8% of FM1 cases, the gold answer string is absent from the Trafilatura extraction of the gold article. FM2 (rank loss) applies when valid text evidence exists but falls outside the reader's top-3 window. FM2 rank compares the two modalities on the FM2 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 16 shows representative SimpleQA examples for each failure mode.

Table 15: Paired failure decomposition on SimpleQA by evidence type. FM1: parser loss; FM2: rank loss. Rows overlap (column sums > Overall).

		Text fail†		FM2 rank			
Type	n	FM1	FM2	PIXELRAG	Trafilatura	PIXELRAG fail	Both 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 16: Representative SimpleQA examples for each failure mode.

Mode	Ev. type	Question (abbreviated)	Answer	PIXELRAG	Trafilatura
FM1	table	How many shots did Inter attempt on target in the CL Final . . . ?	7	1	>100
FM1	para.	What disease was Elizabeth Esteve-Coll diagnosed with . . . ?	multiple sclerosis	1	>100
FM1	list	Which award did Reza Aslan receive in 2014?	Intersections Int'l Award	2	>100
FM2	table	On what day was Javier Zanetti's first daughter born?	11 June 2005	2	89
FM2	para.	What was the rate of climb of the Grumman F4F-3 Wildcat . . . ?	11.70 m/s	2	91
FM2	list	Who sketched the Taddei Tondo following its arrival at the RA . . . ?	John Constable	1	62
FM2	infobox	What team finished with 38 pts in the 2021–22 PL season?	Leeds United	1	68

Reader error. Reader error (FM3) 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, FM3 is 7.0% (56 of 799 questions with valid PIXELRAG evidence in top-3).

Figures 10–13 show fully expanded case studies (one for FM1, two for FM2, one for FM3), 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 (FM2) cases in Table 15, 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 1.

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
Soccernet.  22 May 2010.  Archived from ...
```

↪ *Bibliography, not data.*

⋮

rank 26, score 0.614 — *2010 UEFA Champions League final* (chunk 4)

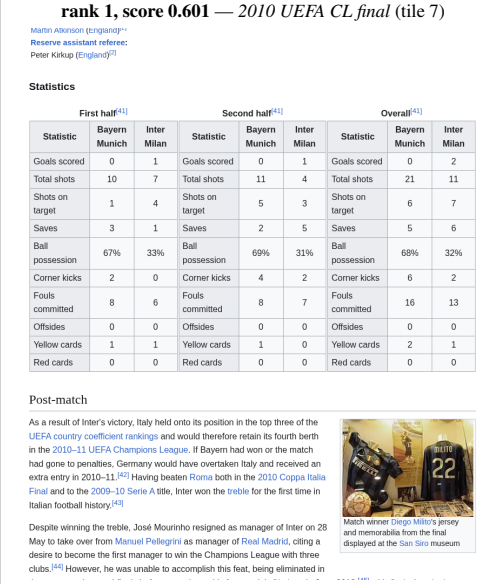
```
Statistics | | | | Post-match As a result of
Inter's victory, Italy held onto its position
...
```

↪ *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)


Martin Atkinson (England)[41]
Reserve assistant referee:
Peter Kirkup (England)[41]

Statistics

First half[41]			Second half[41]			Overall[41]		
Statistic	Bayern Munich	Inter Milan	Statistic	Bayern Munich	Inter Milan	Statistic	Bayern Munich	Inter Milan
Goals scored	0	1	Goals scored	0	1	Goals scored	0	2
Total shots	10	7	Total shots	11	4	Total shots	21	11
Shots on target	1	4	Shots on target	5	3	Shots on target	6	7
Saves	3	1	Saves	2	5	Saves	5	6
Ball possession	67%	33%	Ball possession	69%	31%	Ball possession	68%	32%
Corner kicks	2	0	Corner kicks	4	2	Corner kicks	6	2
Fouls committed	8	6	Fouls committed	8	7	Fouls committed	16	13
Offsides	0	0	Offsides	0	0	Offsides	0	0
Yellow cards	1	1	Yellow cards	1	0	Yellow cards	2	1
Red cards	0	0	Red cards	0	0	Red cards	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.[42] 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.[43]

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.[44] However, he was unable to accomplish this feat, being eliminated in

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

⋮

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

Figure 10: **FM1 / 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.

- **"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 FM2 rank is 22.5 (vs. 16–18 for other evidence types). In these cases, PIXELRAG retrieval finds the answer at mean rank 1.6 on the same questions.

Quantifying PIXELRAG's immunity to infobox displacement. We identify the 42 questions where (1) paragraph-type answer-bearing evidence exists but ranks outside text retrieval's top-3, and (2) text retrieval places an infobox chunk at rank 1. On this subset, PIXELRAG retrieves valid evidence (any type) in top-3 for 37 of 42 questions (88%), and specifically recovers the paragraph evidence in top-3 for 20 of 42 (48%). The visual embedding model is not displaced because the infobox tile has a distinct bordered-sidebar layout (key-value pairs in a colored box) that is visually dissimilar from body paragraphs (flowing text under section headings), even though the linearized text of both produces similar embedding vectors due to shared entity mentions.

Table 17 decomposes failures by pipeline stage for both text and PIXELRAG retrieval across evidence types. Table 18 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.

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 -- Boletin Oficial del
Estado ...  Gibson, Ian (1997) pp. 603–604
...
```

↪ *References section, no biographical content.*

⋮

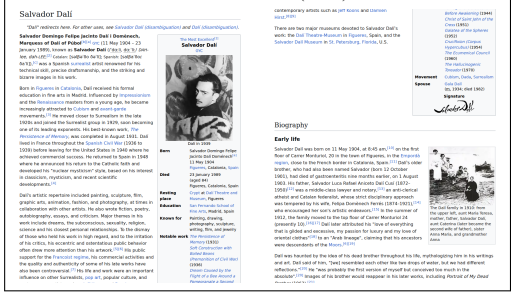
rank 12 — *Salvador Dalí* (chunk 1)

```
...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"
...
```

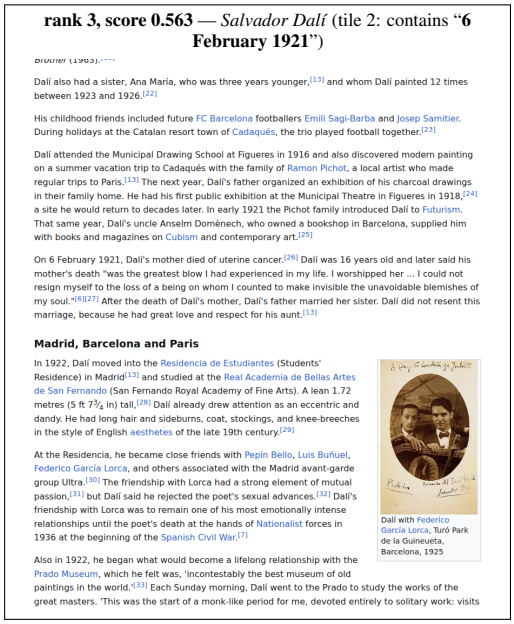
Text reader: "…the specific day and month of Salvador Dalí's mother's death are not mentioned; only the year 1921 is given." ✗

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**")



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

Figure 11: **FM2 / rank loss (paragraph evidence).** Text retrieval finds the correct article but ranks the entity infobox at the top — which lists Dalí's own birth/death, not his mother's — displacing the answer-bearing body paragraph to rank 12. The visual embedding distinguishes the infobox sidebar from the body section, surfacing the relevant tile in the top-3.

Table 17: SimpleQA failure causes by evidence type for text vs. PIXELRAG retrieval (Qwen3.5-4B reader, top-k=3). Rows overlap. Unlike the type-specific evidence recall of Table 2, Any-ev. R@3 counts any valid evidence item, and counts come from the failure-analysis run of Appendix D.4.

Modality	Evidence	n	Failures	Doc miss	Ev. miss	Reader res.	Any-ev. R@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

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.*

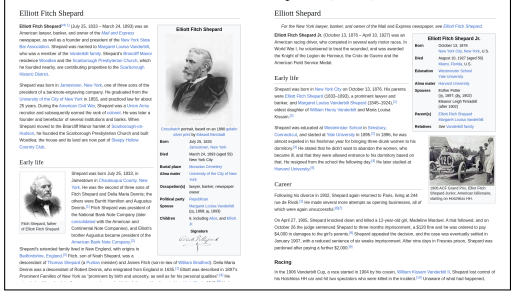
⋮

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 ...`

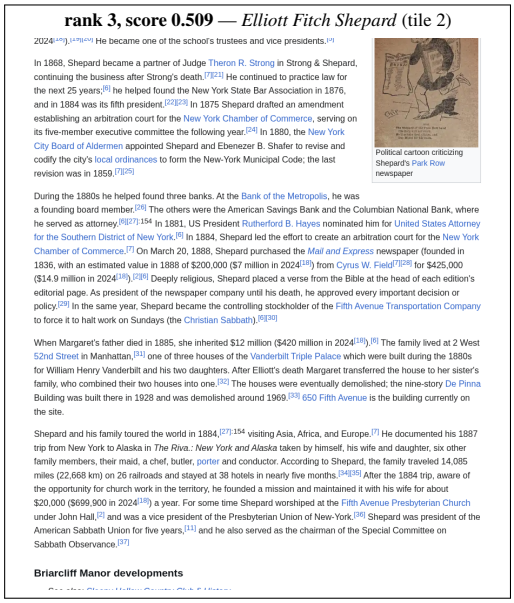
Text reader: "…there is no information stating that Elliott Fitch Shepard was nominated as United States Attorney …" ×

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)



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

Figure 12: **FM2 / 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 recovers the lead tile (which contains the same text) at rank 1.

Table 18: SimpleQA evidence-tile rank distribution by evidence type (top-100 retrieval, Qwen3.5-4B reader). Rows overlap.

Modality	Evidence type	n	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: "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.*

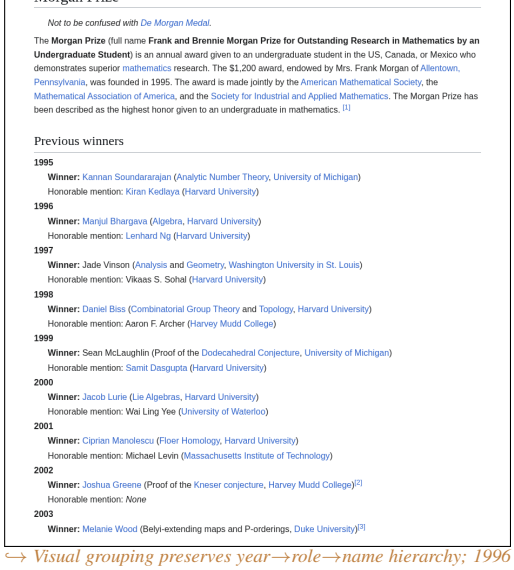
⋮

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.*

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: Vikaas S. Sohal (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.*

⋮

Pixel reader: "**Lenhard Ng** (Harvard University)" ✓

Figure 13: **FM3 / reader error.** 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.

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 linearizes HTML into flat strings, destroying table and list structure; this baseline instead feeds the reader *raw HTML* with intact `<table>`, `<ul>`, 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 (Kiwix Association, 2007) 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 (Qwen3.5-4B, `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 ($3.8\times$) 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 $\sim$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.

Results. Table 19 reveals a striking dissociation between retrieval and reading when HTML is used end-to-end:

- **With HTML input, reading fails even where retrieval does not.** Where HTML-RAG retrieves as well as or better than Trafilatura (Recall@3 +2.1 pp on NQ, +1.0 pp on NQ-Tables), its accuracy still falls by 29.4 and 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.

Table 19: 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.

	NQ		NQ-Tables		SimpleQA		MMSearch	EVQA	
Method	R@3	Acc	R@3	Acc	R@3	Acc	Acc	R@3	Acc
Trafilatura (text)	45.8	55.9	37.2	42.5	77.4	71.6	24.7	6.4	29.6
HTML-RAG	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	−7.8	−11.8	−2.4	−4.9	+5.1

- **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.

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 ($\sim$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.

D.7 Tables Split Across Tiles

PixelRAG improves NQ-Tables Recall@3 from 37.2% to 51.1% and accuracy from 42.5% to 48.8% over Trafilatura (Table 1), despite fixed tiling: non-overlapping 1024-pixel tiles can separate an answer cell from the headers needed to interpret it, and adjacent tiles are embedded independently, so they are not guaranteed to be retrieved together. To measure how often this happens, we re-render each NQ-Tables gold page, read the gold table's pixel box from the DOM, and count the 875×1024 tiles it spans. Of 756 locatable gold tables, 57% exceed one tile in height and 68.8% are split across two or more tiles once boundary straddling is counted; the median gold table is 1,145 px tall (p90 4,032 px). Accuracy degrades with span (Table 20), as the relevant rows increasingly fall outside the reader's top-3 tiles. This is a major source of PixelRAG's errors on table evidence in Table 17, where failures concentrate in ranking rather than coverage; text chunking splits the same tables from their headers, so neither modality solves it. Neighbor-tile expansion and table-aware tiling are concrete directions for improvement.

Table 20: NQ-Tables answer accuracy by the number of 875×1024 tiles the gold table spans (n=756 locatable gold tables).

Tiles spanned	Share of tables	EM
1	31.2%	28.4
2	42.5%	27.4
3+	26.3%	23.1

D.8 Rendering Success on the News Corpus

Table 21: Crawl and render accounting for the news corpus. Download failures are HTTP 404s; every downloaded page rendered into the index.

Source	Attempted	Download failed	In index
BBC	356,358	409 (0.11%)	355,949
AP News	261,241	0 (0%)	261,241
CNN	49,924	7 (0.01%)	49,917
Total	667,523	416 (0.06%)	667,107

Table 21 reports per-source crawl and render accounting from the pipeline database for the news corpus, where pages are dynamic and heterogeneous. Of 667,523 attempted pages, 416 (0.06%) fail to download, all with HTTP 404, and every downloaded page renders into the index after retries. The news corpus uses neither the element stripping of Appendix A.1 nor any Wikipedia-specific processing, and PIXELRAG (base) still improves LiveVQA accuracy from 59.0% to 70.3% and Recall@3 from 16.2% to 38.9% without news-domain training (Table 1).

E LIMITATIONS 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. The embedding index is also larger than its text counterpart (30M tiles vs. 15.7M chunks; 120 GB vs. 64 GB in fp16), which modestly increases retrieval latency (end-to-end p50 of $\sim$1.9 s vs. $\sim$1.4 s; Appendix C.8).

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. Our corpora are also limited to Wikipedia and three news outlets; heterogeneous open-web crawls (e.g., CommonCrawl-scale pages with dynamic content) remain to be tested.

A fourth limitation is fixed-height tiling: a large table can be split across tiles, separating answer cells from their headers, and this is a major source of PIXELRAG's errors on table evidence, which concentrate in ranking rather than coverage (Appendix D.7). Neighbor-tile expansion and table-aware tiling are natural mitigations.

Screenshots are a retrieval representation, not a replacement for accessible HTML: visual embeddings may be weaker than lexical retrieval for exact strings, rare identifiers, and copy-sensitive tasks, and the pixels alone do not expose machine-readable metadata such as DOM roles, alt text, timestamps, or link targets (the sidecar-metadata mitigation above applies to these as well).

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.



ticket and roster spot.[^]

Cal had outscored their opponents on the season, 312–33, but the W&J defense held the Golden Bears' potent offense, led by Brick Muller, with no points and no completed passes, just two first downs, and only 49 yards rushing.[4][5] In one of the most disputed plays in Rose Bowl history, a W&J rushing touchdown was overturned for an offside violation.[4] The game was notable as the last time a "small school" would be represented in the Rose Bowl. W&J's team featured three Rose Bowl firsts: Herb Kopf was the first freshman to play, Charles Fremont West was the first African-American to play quarterback,[7] and Hal Erickson became the only man ever to play in two Rose Bowls, with two teams, without losing.[1] W&J's Russ Stein was named Player of the Game.[8][9] He was inducted into the Rose Bowl Hall of Fame in 1991.[9] The Red and Black finished the season with a share of the "mythical national championship," as determined by the Boand System.

Records

- The only scoreless tie in Rose Bowl history, and one of the four scoreless ties in major bowl games.[8]:56
- Fewest passing yards: 0[8]:50
- Tied the record for fewest total points[8]:52

References

1. ^ a b Campbell, Jim (August 2006). "The 1922 Rose Bowl: David v. Goliath" (PDF). *College Football Historical Society Newsletter*. LA84 Foundation. Archived from the original (PDF) on March 26, 2009. Retrieved February 7, 2012.
2. ^ "GRANDSON OF 1924 GRADUATE DONATES SCRAPBOOK OF ROSE BOWL MEMORIES". Washington & Jefferson College. April 10, 2014.
3. ^ Florence, Mal (January 1, 1998). "Some Rose Bowl Stories Have a Different Spin". *Los Angeles Times*. Retrieved April 26, 2010.
4. ^ a b c d e f g h E. Lee, North (1991). "Chapter 8: A Season to Remember". *Battling the Indians, Panthers, and Nittany Lions: The Story of Washington & Jefferson College's First Century of Football, 1890-1990*. Daning Books. pp. 97–107. ISBN 978-1-878302-03-8. OCLC 24174022.
5. ^ "Robert Murphy : Washington and Jefferson College Football". Archived from the original on October 7, 2010. Retrieved October 27, 2010.
6. ^ "1922 Rose Bowl". *rosebowlhistory.org*. Archived from the original on October 27, 2016.
7. ^ "Charles "Prune" West (1922)". *U. Grant Miller Library Digital Archives*. Washington & Jefferson College. Archived from the original on May 25, 2024. Retrieved January 28, 2021.
8. ^ a b c d "Bowl/All-Star Game Records" (PDF). National Collegiate Athletic Association. 2009. Retrieved June 9, 2010.
9. ^ "Rose Bowl Hall of Fame". Archived from the original on August 16, 2010. Retrieved April 14, 2008.

External links

- Media related to 1922 Rose Bowl at Wikimedia Commons

Rose Bowl Game

History & conference tie-ins · History · Rose Bowl (stadium) · College Football Playoff (New Year's Six) · Broadcasters ·

Input: rendered Wikipedia tile

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.