Prism: A Fully Local Multi-Modal RAG System and the Three Expectations Measurement Disproved

Dr. M K Jayanth Kannan[1], Himanshu[2], Tejashri Choudhary[3],
Aditya Gupta[4], Ayush Yadav[5], Akshat Jha[6], Ayushi Chouhan[7]
[1]*Professor, School of Computing Science and Engineering,*
VIT Bhopal University, Bhopal–Indore Highway, Kothrikalan, Sehore, Madhya Pradesh – 466114.
[2,3,4,5,6,7] *B.Tech Students, School of Computing Science and Engineering,*
VIT Bhopal University, Bhopal–Indore Highway, Kothrikalan, Sehore, Madhya Pradesh – 466114.

***Abstract*—Enterprise knowledge is fragmented across scanned PDFs, photographs, slide decks, spreadsheets, recordings, and code, and conventional search requires the user to already know a filename. This paper presents Prism, a multi-modal retrieval-augmented generation (RAG) system that retrieves files by described intent and runs entirely on a single laptop with no external service. Prism ingests ten file formats through an event-driven pipeline, indexes text and images in separate embedding spaces, fuses dense, lexical, and vision-language retrievers by reciprocal rank, reranks with a cross-encoder, and produces grounded answers with citations that resolve to a page or a timestamp. Sensitive identifiers are detected using format checksums and masked in a second payload that is the only text a language model ever sees. On a 52-query labelled benchmark, the full pipeline reaches Precision@1 of 0.962 and Recall@5 of 1.000 at a median retrieval latency of 91.5 ms. The measurements also disproved three expectations. Hybrid fusion reduced top-1 precision relative to BM25 alone (0.788 versus 0.885), and only cross-encoder reranking, which consumes 85% of the latency budget, converted the recall it gained back into precision. The vision-language retriever changed the top result on only one of five cross-modal queries, because optical character recognition (OCR) had already made those images findable as text. And an OCR engine reporting 0.99 confidence returned text with every inter-word space removed, leaving dense retrieval degraded and lexical retrieval unable to match a single query term. These negative findings are reported alongside the positive results, together with what the benchmark cannot support.**

***Index Terms*—Multi-Modal Retrieval, Retrieval-Augmented Generation, Hybrid Search, Reciprocal Rank Fusion, Cross-Encoder Reranking, Optical Character Recognition, PII Redaction, On-Device Inference, Enterprise Knowledge Search.**

I. INTRODUCTION

A finance manager looking for last March's invoice above ten thousand dollars does not know it is called INV-2024-0312.pdf. Someone looking for a photograph of their identity card does not know it is IMG_2039.jpg. Conventional file search answers a question the user cannot ask: it matches filenames, folder paths and literal keywords, while the information that would identify the file is inside it as pixels in a scan, as speech in a recording, or as a paraphrase the keyword index never saw. Retrieval-augmented generation [1] addresses the retrieval half of this problem by grounding a language model in a corpus at query time, and a substantial literature now covers dense retrieval [2], hybrid dense–sparse search [3], and vision-language document retrieval [4]. Much of that work, however, is evaluated on public text benchmarks, deployed as a cluster of services, and reported through aggregate quality metrics that do not say which component earned the result. The scientific rationale behind Prism lies in three constraints that sharpen rather than weaken the engineering questions. Firstly, one machine and no external service: every model runs locally on a 16 GB Apple silicon laptop; there is no API key, and no document or query leaves the host. Secondly, no container runtime: the host has no Docker daemon, which eliminates every vector database normally deployed as a server and forces an embedded storage design. Thirdly, every reported number is emitted by the running system: the pipeline writes a timed event per stage and a full record per query, and the evaluation harness and the interface read the same tables. The main contributions of this

work are: A complete multi-modal ingestion and retrieval pipeline spanning ten formats, with per-file event-driven updates and citations that resolve to a page number or a media timestamp (Sections III–VI). A component ablation over a 52-query labelled set showing that hybrid fusion alone is a regression against BM25 on top-1 precision, and that a cross-encoder reranker is what makes fusion worthwhile (Section IX). A measured account of when vision-language retrieval does and does not contribute, including the finding that it changed the top result on only one of five cross-modal queries on a corpus whose images carry recoverable text (Section X). An OCR failure mode high reported confidence with all whitespace removed that damages lexical and dense retrieval asymmetrically, with a repair pass and before/after retrieval measurements (Section X). A two-payload redaction design in which raw text remains indexed for retrieval while a checksum-validated masked copy is the only text a language model receives (Section XI). What the evaluation does not establish is also stated plainly (Section XI): the corpus is 24 seeded files, it is synthetic, and generation faithfulness is spot-checked rather than scored.

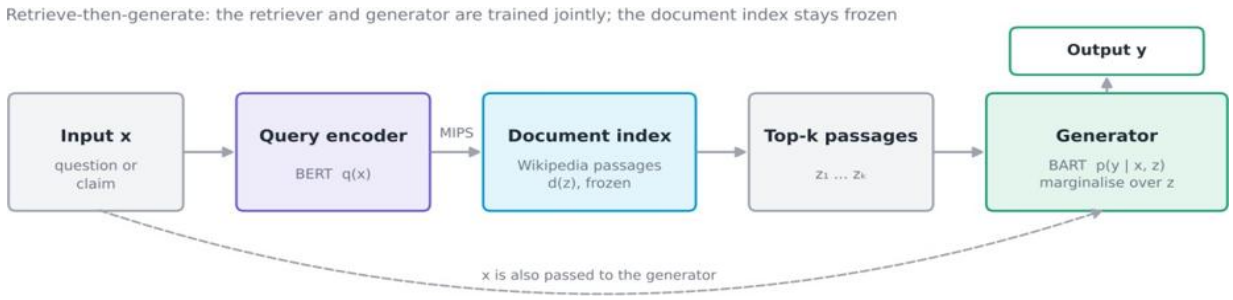
II. LITERATURE REVIEW OF EXISTING SYSTEMS

The literature relevant to Prism spans retrieval-augmented generation, dense, sparse and hybrid retrieval, multi-modal embeddings, vision-native document retrieval, speech and document parsing, and privacy in retrieval systems. RAG conditions a generator on passages retrieved at inference time [1], decoupling the knowledge a system can use from the knowledge frozen in its weights; Fusion-in-Decoder [5] encodes each passage separately and lets the decoder attend across all of them, and recent surveys [6] organise the design space into pre-retrieval, retrieval and post-retrieval stages. Dense bi-encoders [2], [7] capture paraphrase but are weak on rare literal tokens; BM25 [8] remains a strong lexical baseline for identifiers, and learned sparse models such as SPLADE [9] narrow the gap from the lexical side. Cross-encoder rankers [10] jointly encode query and passage for high precision at high cost. Contrastively trained text encoders such as E5 [11] and BGE [12] provide strong retrieval quality at modest size, with BGE requiring an instruction prefix on the query side only. HNSW [14] underlies most production approximate nearest-neighbor indexes, including the one used here [15]. Whisper [17], run through its CTranslate2 reimplementation [18], provides multilingual transcription with segment timestamps, and PP-OCR [19] is a compact OCR pipeline. On the privacy side, naive redaction before embedding is known to distort the vector space; the seed technical report for this project [20] describes a sanitized-payload pattern which Prism implements, with format checksums [21], [22] supplying the precision that pattern matching alone lacks. Six representative studies are reviewed below in terms of objective, technology, methodology, efficiency, and issues (Figures 1–6).

2.1 Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

Paper 1	Objective used	Technology used	Methodology used	Efficiency	Issues
Title: Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks Authors: P. Lewis et al. Venue: NeurIPS Year: 2020 arXiv: 2005.11401 URL:	1. Combine Memories: Give a seq2seq generator access to an explicit, non-parametric document index alongside its parametric knowledge. 2. Improve Factuality: Produce more specific and factual output on knowledge-intensive tasks such as open-domain QA and fact verification. Updatable Knowledge:	1. Dense Passage Retriever: BERT bi-encoder used as the retriever. 2. BART Generator: Pre-trained seq2seq model conditioned on retrieved passages. FAISS Index: Maximum-inner-product search over a Wikipedia	1. Retrieval: Fetch the top-k passages for each input by inner product. 2. Marginalization : RAG-Sequence uses one passage per output sequence; RAG-Token can switch passage per token. Training: Query encoder and generator fine-tuned end to end;	**1.** Accuracy: State-of-the-art results on several open-domain QA benchmarks at publication. **2.** Quality: Generations judged more specific and factual than a BART-only baseline. Maintainabilit	1. Text Only: A single Wikipedia text index; no images, scans or audio. 2. Memory Cost: A large dense index must be held for search. 3. Attribution: No validation that generated statements are backed by a cited passage.

https://arxiv.org/abs/2005.11401 Ref: [1]	Allow world knowledge to be changed without retraining the model.	dump split into short passages.	document encoder kept frozen so the index is not rebuilt.	y: Swapping the index updates knowledge without retraining.	Domain: Evaluated on public benchmarks rather than enterprise documents.



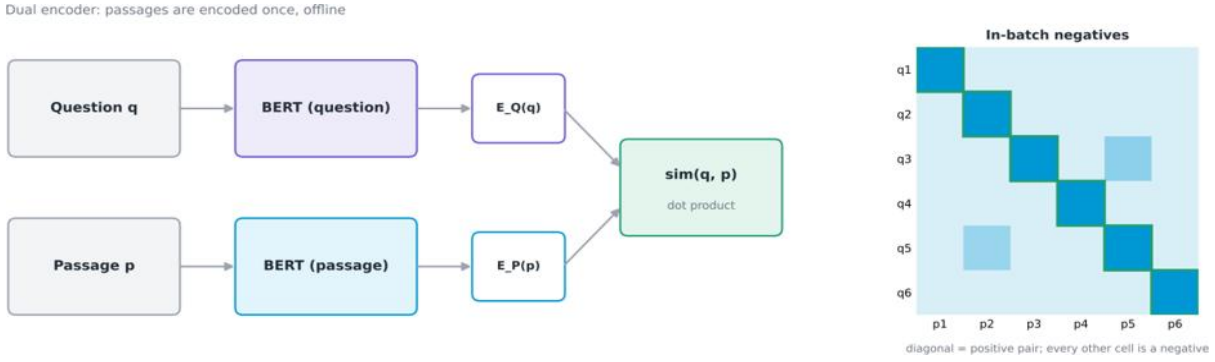
[1]Retrieve-then-generate architecture of RAG: a query encoder retrieves top-k passages from a frozen index by maximum inner-product search, and the generator marginalizes over them. Schematic redrawn by the authors from [1].

Figure 1: Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks Literature Review

"Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" [1] introduces the retrieve-then-generate architecture on which Prism is built.

2.2 Dense Passage Retrieval for Open-Domain Question Answering

Paper 2	Objective used	Technology used	Methodology used	Efficiency	Issues
Title: Dense Passage Retrieval for Open-Domain Question Answering Authors: V. Karpukhin et al. Venue: EMNLP Year: 2020 Pages: 6769–6781 DOI: 10.18653/v1/2020.emnlp-main.550 URL: https://aclanthology.org/2020.emnlp-main.550/ Ref: [2]	**1.** Dense over Sparse: Show that learned dense embeddings alone can outperform TF-IDF/BM25 retrieval for open-domain QA. **2.** Simple Training: Learn the retriever from question–passage pairs without additional pre-training.	**1.** Dual BERT Encoders: Separate question and passage encoders. **2.** Dot-Product Similarity: Relevance as inner product of the two vectors. **3.** FAISS: Approximate nearest-neighbour search over pre-computed passage vectors.	**1.** Negatives: In-batch negatives plus BM25-mined hard negatives. **2.** Offline Indexing: Passages encoded once, queries encoded at search time. **3.** Evaluation: Top-k retrieval accuracy on several QA datasets, then end-to-end QA.	**1.** Retrieval Gain: Large absolute improvements in top-20 accuracy over a Lucene BM25 baseline. **2.** End-to-End QA: Better downstream answer accuracy. **3.** Speed: Fast retrieval through ANN search.	**1.** Literal Tokens: Weaker where exact lexical overlap matters. **2.** Supervision: Needs labelled question–passage pairs. **3.** Rare Identifiers: Invoice numbers and codes are poorly represented. **4.** Modality: Text only.



Dense Passage Retrieval: separate question and passage encoders scored by dot product, trained with in-batch negatives. Schematic redrawn by the authors from [2].

Figure 2: Dense Passage Retrieval for Open-Domain Question Answering Literature Review

"Dense Passage Retrieval for Open-Domain Question Answering" [2] established the dual-encoder approach used by Prism's semantic retriever. The key objectives of the study include: Demonstrating that dense embeddings can replace sparse retrieval for open-domain QA, and that a strong retriever can be trained from question–passage pairs alone. The technology used in the study includes: Two BERT encoders, dot-product similarity and FAISS approximate nearest-neighbour search. The methodology integrates: Training with in-batch and BM25-mined hard negatives, offline encoding of the passage collection, and evaluation by top-k retrieval accuracy followed by end-to-end QA. Cybersecurity, authentication, cloud computing, and machine learning: Research on cybersecurity has examined threats and countermeasures across networked systems, cloud platforms, and user authentication mechanisms. The work on protecting confidential data over distributed cloud storage [28] addresses the importance of safeguarding sensitive information in distributed environments, while the book on network security attacks and countermeasures [29] provides a broader foundation for understanding attack models and defensive strategies. Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods

Paper 3	Objective used	Technology used	Methodology used	Efficiency	Issues
Title: Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods Authors: G. V. Cormack, C. L. A. Clarke, S. Buettcher Venue: ACM SIGIR Year: 2009 Pages: 758–759 DOI: 10.1145/1571941.1572114 Ref: [3]	1. Simple Fusion: Propose an unsupervised way to merge the ranked lists of several retrieval systems. 2. Comparison: Test it against Condorcet fusion and learned rank-combination methods.	1. RRF Formula: Each document scores the sum of 1/(k + rank) over all input rankings, with k = 60. 2. TREC Runs: Submitted system rankings as inputs. 3. Learning-to-Rank Baselines: Supervised rank learners for comparison.	1. Fusion Experiments: Combine multiple TREC system runs with RRF and competing methods. 2. Evaluation: Compare mean average precision against the best individual systems and fusion baselines.	**1.** Effectiveness: RRF matched or outperformed the competing fusion and learning methods. **2.** No Training: Requires no labelled data. **3.** No Normalisation: Uses ranks only, so incomparable score scales do not matter.	1. Lost Confidence: Score magnitudes are discarded. 2. Fixed Constant: k is chosen empirically. 3. Equal Voting: A confident minority retriever is averaged against indifferent ones. 4. Short Paper: Two-page study.



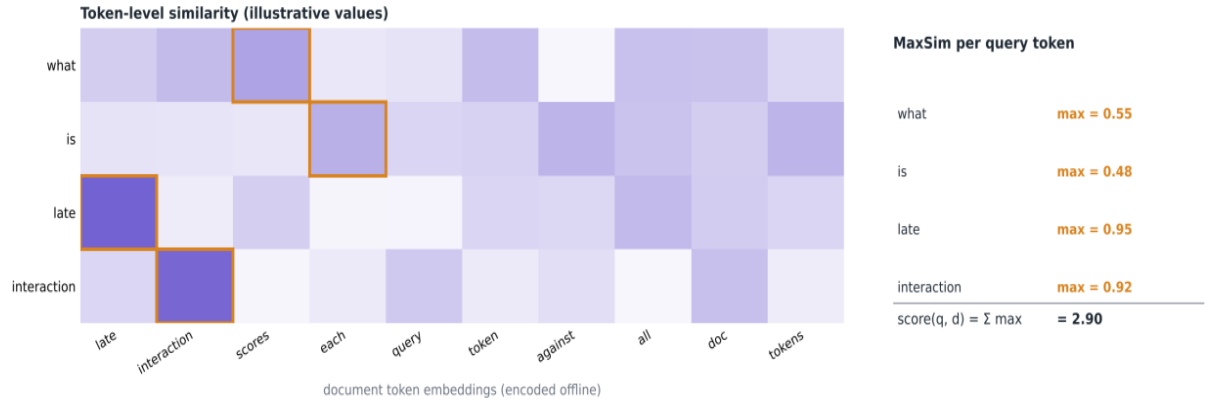
Reciprocal rank fusion of three ranked lists with k = 60, computed on a worked example using Prism's three retrievers. Note how receipt.jpg, ranked firs]t by CLIP only, ends a close second. Illustration by the authors after [3].

Figure 3: Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods Literature Review

"Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods" [3] defines the fusion formula Prism uses to merge its three retrievers. The key objectives of the study include: Proposing a simple, unsupervised rank-fusion method and comparing it with Condorcet fusion and supervised rank learning. Image-based authentication using randomized selection [30] and the performance analysis of bag-of-password authentication across Python, Java, and PHP implementations [31] explore approaches to improving authentication security and implementation efficiency. Research on cybercrime

and open-source software [32], strategic information-security planning using decentralized identity mechanisms [33], and machine-learning techniques for malware detection [34] further highlight the need for layered security, proactive threat detection, and user-oriented protection.ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT

Paper 4	Objective used	Technology used	Methodology used	Efficiency	Issues
Title: ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT Authors: O. Khattab, M. Zaharia Venue: ACM SIGIR Year: 2020 Pages: 39–48 DOI: 10.1145/3397271.3401075 Ref: [16]	1. Efficient BERT Ranking: Keep BERT-level ranking quality at a fraction of the cost of a cross-encoder. 2. End-to-End Retrieval: Support retrieval from a large collection, not only reranking.	**1.** BERT Encoders: Produce one contextual vector per token. **2.** Late Interaction: MaxSim operator between query and document token vectors. **3.** FAISS: Vector-similarity index for candidate generation.	1. Independent Encoding: Documents encoded offline; queries encoded at search time. 2. Scoring: For each query token, take the maximum similarity over document tokens and sum. 3. Evaluation: Passage ranking benchmarks including MS MARCO.	**1.** Quality: Competitive with BERT cross-encoder rerankers. **2.** Speed: Around two orders of magnitude faster, with about four orders of magnitude fewer FLOPs per query.	1. Storage: One vector per token makes the index far larger than single-vector indexes. 2. Complexity: More involved indexing and search pipeline. . Modality: Text only.



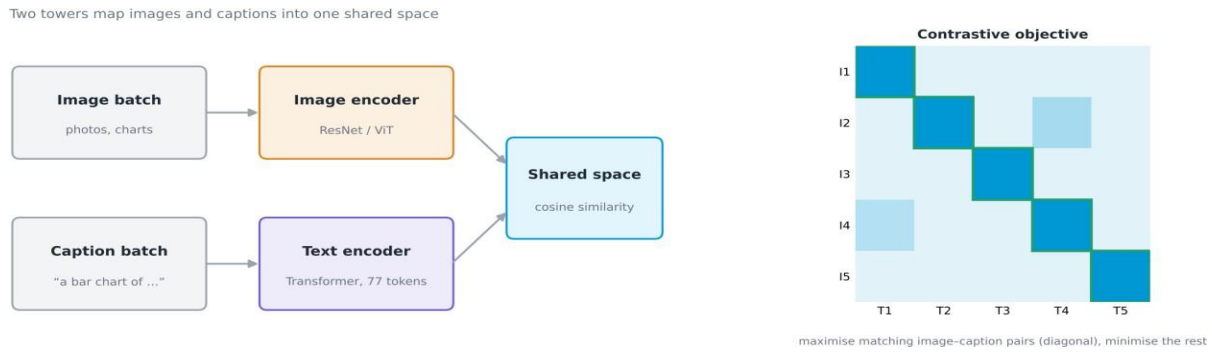
ColBERT late interaction: each query token takes its maximum similarity over all document tokens and the maxima are summed. Values are illustrative. Schematic redrawn by the authors from [16].

Figure 4: ColBERT Literature Review

ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT" [16] introduced late interaction, the scoring mechanism that later vision-native retrievers inherit. The key objectives of the study include: Retaining BERT-level ranking quality at much lower cost, and enabling end-to-end retrieval rather than reranking only. The technology used in the study includes: BERT encoders that emit per-token vectors, the MaxSim late-interaction operator and a FAISS index. The methodology integrates: Offline document encoding, per-query MaxSim scoring summed over query tokens, and evaluation on passage-ranking benchmarks.The study ensures efficiency through: Quality competitive with cross-encoders while running far faster and with far fewer FLOPs per query. Despite its strengths, the study identifies challenges such as: Much larger per-token indexes, a more complex pipeline, and text-only scope. Relevance to Prism: Prism uses a single-vector index plus a cross-encoder on a 20-candidate window instead; late interaction over page images is identified as future work (Section XIII).

2.3 Learning Transferable Visual Models from Natural Language Supervision (CLIP)

Paper 5	Objective used	Technology used	Methodology used	Efficiency	Issues
Title: Learning Transferable Visual Models From Natural Language Supervision Authors: A. Radford et al. Venue: ICML (PMLR 139) Year: 2021 Pages: 8748–8763 arXiv: 2103.00020 Ref: [13]	**1.** Language Supervision: Learn visual concepts from natural-language captions instead of fixed labels. **2.** Zero-Shot Transfer: Apply the model to new tasks without task-specific training.	**1.** Image Encoder: ResNet and Vision Transformer variants. **2.** Text Encoder: Transformer with a 77-token context. **3.** Contrastive Objective: Image–text matching on a large web-collected dataset.	**1.** Pre-training: Match each image with its correct caption within a batch. **2.** Zero-Shot Classification: Embed class-name prompts and pick the closest. **3.** Evaluation: Benchmarked across many vision datasets.	**1.** Zero-Shot Accuracy: Matched a fully supervised ResNet-50 on ImageNet without using its labelled training set. **2.** Robustness: Better behaviour under distribution shift than comparable supervised models.	**1.** Short Text: 77-token text encoder is weak for document passages. **2.** Fine-Grained Tasks: Struggles with counting and abstract tasks. **3.** Bias: Inherits social biases from web data.



CLIP: image and text encoders trained contrastively so that matching image–caption pairs score highest. Schematic redrawn by the authors from [13].

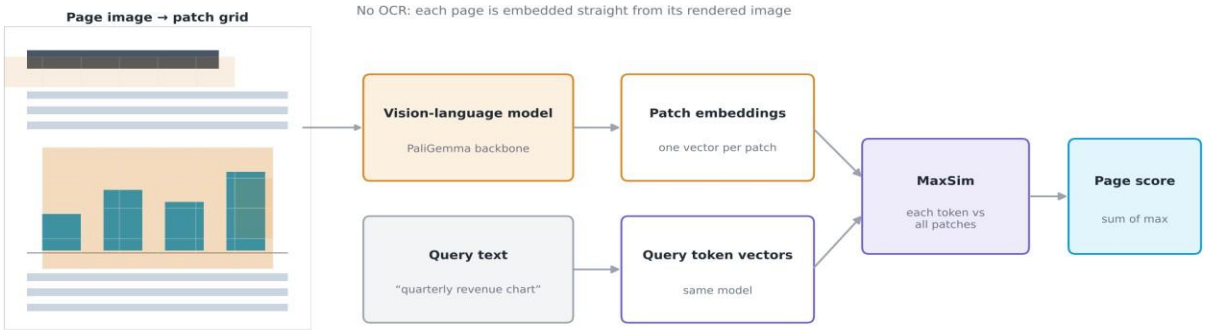
Figure 5: Learning Transferable Visual Models from Natural Language Supervision (CLIP) Literature Review

"Learning Transferable Visual Models from Natural Language Supervision" [13] introduced CLIP, which Prism uses for its image index and text-to-image search. The key objectives of the study include: Learning visual concepts directly from natural-language supervision and transferring them zero-shot to new tasks. The technology used in the study includes: A ResNet or ViT image encoder, a Transformer text encoder with a 77-token context, and a contrastive image–text objective. The methodology integrates: Contrastive pre-training on image–caption pairs, zero-shot classification through text prompts, and evaluation across many datasets. The study ensures efficiency through: Zero-shot ImageNet accuracy matching a supervised ResNet-50 and improved robustness to distribution shift. Despite its strengths, the study identifies challenges such as: A short text context poorly suited to document passages, weakness on fine-grained and abstract tasks, and inherited bias. Relevance to Prism: This motivated Prism's two-space design: forcing document text through CLIP gave P@1 0.269 against 0.769 for the dedicated text encoder (Table I).

2.4 ColPali: Efficient Document Retrieval with Vision Language Models

Paper 6	Objective used	Technology used	Methodology used	Efficiency	Issues
Title: ColPali: Efficient Document Retrieval with Vision Language Models Authors: M. Faysse et al. Venue: arXiv preprint	**1.** OCR-Free Retrieval: Retrieve document pages directly from their rendered images. **2.** Benchmark:	**1.** Vision-Language Model: PaliGemma-based page encoder. **2.** Late Interaction:	**1.** Page Embedding: Each page image becomes a set of patch vectors. **2.** Scoring: Query tokens matched to	**1.** Accuracy: Outperforms text-extraction pipelines on visually rich documents. **2.** Indexing Speed:	**1.** Storage: Many vectors per page. **2.** Compute: A multi-billion-parameter model is costly on a

Year: 2024 arXiv: 2407.01449 URL: https://arxiv.org/abs/2407.01449 Ref: [4]	Introduce a benchmark for visually rich document retrieval (ViDoRe).	ColBERT-style multi-vector scoring over image-patch embeddings.	patches by MaxSim. 3. Comparison: Against OCR/text pipelines, captioning pipelines and single-vector models.	Skips OCR and layout analysis. 3. Simplicity: End-to-end trainable.	laptop. 3. Latency: Multi-vector scoring is slower than single-vector search.



ColPali: a vision-language model embeds page-image patches directly, and query tokens are scored against patches by MaxSim, with no OCR step. Schematic redrawn by the authors from [4].

Figure 6: ColPali Literature Review

"ColPali: Efficient Document Retrieval with Vision Language Models" [4] applies late interaction to document images, bypassing OCR entirely. The key objectives of the study include: Retrieving pages from their rendered images without OCR, and introducing a benchmark for visually rich document retrieval. The technology used in the study includes: A PaliGemma-based vision-language encoder with ColBERT-style late interaction over patch embeddings. The methodology integrates: multi-vector page embedding, MaxSim scoring, and comparison with OCR-based, captioning-based, and single-vector pipelines. The study ensures efficiency through: Higher retrieval quality than text-extraction pipelines on visually rich documents, faster indexing, and an end-to-end trainable design. Despite its strengths, the study identifies challenges such as heavy multi-vector storage, high compute for a laptop deployment, and slower scoring than single-vector search. Relevance to Prism: Prism did not implement late interaction; it measured the single-vector page-image baseline beneath it, which reached P@1 0.71 and Recall@5 1.00 on OCR-dependent queries (Section X).

2.5 Research Gap

Across the reviewed studies, retrieval is evaluated on public text benchmarks, deployed as multi-service infrastructure, and reported through aggregate metrics that do not attribute quality to individual components. None of them combines many enterprise file formats, fully on-device inference without a container runtime, checksum-validated redaction, and a per-component ablation in a single system. Prism addresses this gap, and its contribution on the retrieval side is not a new method but a measurement: on the Prism corpus, the usual ordering of these components' importance does not hold.

III. PROPOSED SYSTEM DESIGN: MULTI-MODAL ENTERPRISE RETRIEVAL

The scientific rationale behind the proposed system is that an enterprise user describes what a file contains, not what it is called, and that the evidence needed to find it lives in pixels, speech and paraphrase as often as in literal keywords. Prism therefore combines three independent retrievers dense semantic search, lexical BM25 and vision-language (CLIP) search and lets a cross-encoder decide among their candidates. Three design properties shape the rest of the system.

- Updates are per file and event-driven. A filesystem event or an HTTP upload enqueues a job in a durable queue; a worker parses, chunks, redacts, embeds and indexes that file alone. Cost is O (changed files), not O(corpus), and there is no

rebuild step. Change is detected by SHA-256 over the file bytes rather than modification time, because a touched-but-unchanged file must not trigger a re-embed the most expensive stage and a byte-changed file with a preserved modification time must not be missed.

- A chunk never spans a citation boundary. The product promises a citation resolving to a page or a timestamp, which constrains chunking before any quality argument applies: no chunk crosses a page, a slide, a heading section or a speaker turn.
- Two embedding spaces, not one. Text is embedded with bge-small-en-v1.5 (384-d) and images with CLIP ViT-B/32 (512-d). A single joint space would force every text chunk through CLIP's 77-token text encoder, which is markedly weaker at document retrieval; the ablation quantifies the gap (P@1 0.269 versus 0.769). The two indexes are fused at retrieval time instead.

Storage under the no-container constraint. The absence of a Docker daemon eliminated pgvector, Weaviate, Milvus and Elasticsearch. Qdrant [15] was selected because its Python client embeds the engine in-process against a local directory while exposing the same API as a clustered server, so the deployment decision is reversible without a code change. The lexical index uses SQLite FTS5 [23], which is compiled into CPython's bundled SQLite and therefore supplies Okapi BM25 with no additional infrastructure. One SQLite file also holds the manifest, chunk store, durable job queue, per-stage telemetry, query log and audit trail, so all state except the vector index shares a transactional boundary.

IV. ARCHITECTURE DIAGRAM OF PRISM

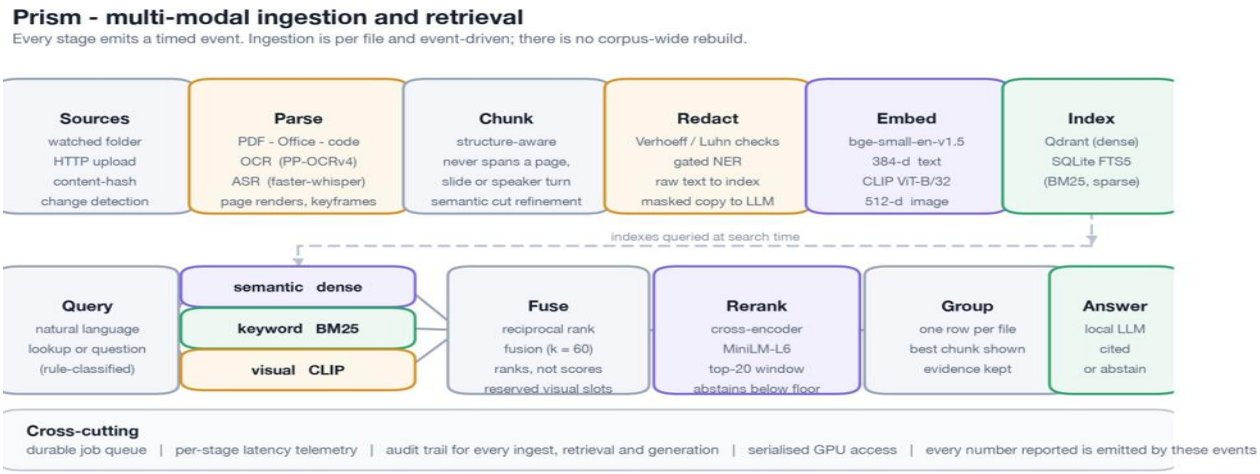


Figure 7: Architecture Diagram of Prism: Multi-Modal Ingestion and Retrieval Pipeline

Ingestion (top row of Figure 7) runs per file and is event-driven: a filesystem event or upload enqueues one durable job, and only that file is parsed, chunked, redacted, embedded and indexed. Retrieval (bottom row) queries three independent retrievers, fuses them by rank, reranks a bounded window with a cross-encoder, and groups the result to one row per file before an optional grounded answer is generated. Cross-cutting services the durable job queue, per-stage latency telemetry, an audit trail for every ingest, retrieval and generation, and serialised GPU access support both halves. Every stage emits a timed event, and the metrics in this paper are read from those events.

V. METHODOLOGY AND ALGORITHMS USED

The methodology employed in Prism involves multiple stages of ingestion, parsing, chunking, redaction, embedding, indexing, retrieval, reranking and generation. The system follows the structured workflow outlined below:

- Data Ingestion: Files arrive through a watched folder or an HTTP upload. Each event enqueues one durable job, and a SHA-256 content hash decides whether the file has actually changed. Parsing: PDFs are parsed with PyMuPDF; a page yielding fewer than 120 extractable characters is treated as a scan and routed to OCR (PP-OCRv4

through ONNX Runtime). The threshold was set by measurement: genuine text pages returned 900–2400 characters and the scanned page returned exactly zero. Every page is also rendered to PNG for thumbnails, CLIP vectors and page-image retrieval. Office formats preserve heading structure, slide boundaries and speaker notes. Audio and video are transcribed with faster-whisper at int8 with segment timestamps preserved verbatim, and video additionally contributes keyframes.

- Chunking: Within a parser block, sentences are packed greedily to a 320-token target with 64-token overlap. Once a chunk reaches 60% of target, a cut is taken early at a local minimum of adjacent-sentence cosine similarity. Semantic similarity is a tie-break rather than the primary splitter because pure semantic chunking degenerates on lists (one-sentence chunks) and on tables (one enormous chunk). ASR segments of 2–8 seconds are merged to the same target while retaining the true start and end of the merged span. Redaction: Checksum-validated detectors and gated named-entity recognition produce a masked copy of each chunk that is stored alongside the raw text (Section XI).
- Embedding and Device Placement: Embedding latency was measured on both available devices (Figure 8). At batch size 1 the CPU is 21% faster than the Metal backend (5.08 ms versus 6.41 ms) because kernel launch and transfer overhead exceed the compute; at batch 64 the GPU is 1.9× faster. A search query embeds one string on the latency-critical path while ingestion embeds in batches of 32, so the system holds two copies of the text encoder (about 260 MB) and places the query encoder on the CPU and the batch encoder on the GPU.

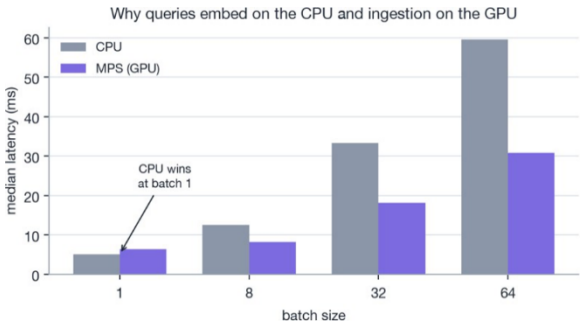


Figure 8: Embedding Latency by Batch Size on CPU and the Apple Metal (MPS) Backend

- Evaluation and Validation: Precision@1, Recall@5, mean reciprocal rank (MRR) and nDCG@10 are computed at file level on a labelled benchmark, ablating one retriever arm at a time against an identical index.

Prism utilizes a combination of retrieval, ranking and generation algorithms:

- Okapi BM25 via SQLite FTS5 [8], [23]: Lexical retrieval with column weights of 1.0 for the chunk body, 2.0 for the section heading and 3.0 for the filename, because a heading or filename hit is a stronger signal for find-my-X queries. Dense Bi-Encoder Retrieval [12]: bge-small-en-v1.5 embeddings searched in Qdrant's HNSW index, with the instruction prefix applied to queries only. CLIP Text-to-Image Retrieval [13]: CLIP's text tower embeds the query and searches the image index of photographs, charts, diagrams, page renders and keyframes. Reciprocal Rank Fusion [3]: The three ranked lists are fused by Equation (1) with k = 60. Ranks are used rather than scores because dense cosine similarity is bounded and stable while BM25 is unbounded and query-dependent; per-query BM25 normalisation was tried first and destabilized the fusion whenever the lexical side returned only one or two hits.

$$\mathrm{RRF}(d) = \Sigma_{r \in R}\ 1 / (k + \mathrm{rank}_r(d)),\ k = 60 \quad (1)$$

- Reserved Image Slots and Surrogates: A text chunk can accumulate RRF credit from two retrievers while an image chunk can accumulate it from only one, which pushed every image below the rerank window; four slots in that window are therefore reserved for the highest-ranked image candidates. Because a cross-encoder scoring an empty string ranks it last, each image is represented to the reranker by a surrogate built from its filename and the OCR text of the same page.
- Cross-Encoder Reranking [10]: ms-marco-MiniLM-L-6-v2 rescores the top 20 candidates. When the entire window falls below a logit of approximately −10.5 the model is reporting that nothing is relevant, and the system falls back not to RRF but to best single-retriever rank, because RRF is a consensus measure and is the wrong tie-break precisely when no consensus exists. Chunks are then grouped to one row per file.

- Grounded Generation [24], [25]: Answers are generated locally by a 4-bit Qwen2.5-3B-Instruct model under MLX, with fallbacks to a Transformers backend and finally to an extractive stitcher, so the product degrades to quoted evidence rather than to an error. Every citation is validated against the passages actually supplied and any unmatched reference is stripped; a dedicated abstention token becomes a distinct interface state.
- Checksum Validation [21], [22]: Verhoeff check digits for Aadhaar-format identifiers and Luhn check digits for payment cards reject candidates that merely look like identifiers.
- OCR Spacing Repair: camelCase splitting, digit/letter boundary splitting and dictionary segmentation restore inter-word spaces lost by OCR (Section X).

VI. PROJECT FUNCTIONAL MODULES IMPLEMENTATION OF PRISM

The development of Prism involves several functional modules, each serving a distinct role in multi-modal ingestion, retrieval and grounded answering:

- File Ingestion Module: Watched-folder and HTTP-upload entry points, a durable job queue, content-hash change detection and a timed event for every stage of every file. Document Understanding Module: Parsers for ten formats native and scanned PDF, DOCX, PPTX, XLSX, CSV, PNG, JPG, MP3, MP4 and source code including OCR, speech recognition, page rendering and video keyframes. Privacy and Redaction Module: Checksum-validated identifier detection, gated named-entity recognition and the two-payload store that keeps raw text for retrieval and a masked copy for generation.
- Indexing and Storage Module: Embedded Qdrant for the text and image vector indexes and a single SQLite file for the FTS5 index, manifest, chunk store, job queue, telemetry, query log and audit trail. Search Module (Find Mode): Three retrievers, reciprocal rank fusion, cross-encoder reranking and one-row-per-file grouping, with each result annotated by the retrievers that matched it and at what rank. Answer Module: Local grounded generation with validated citations that resolve to a page number or media timestamp, and an explicit abstention state.
- Cross-Format Visual Search Module: Text-to-image search and image-to-image nearest-neighbor search over the CLIP index. Interface and Telemetry Module: One color per retriever used consistently across result chips, the latency strip and the architecture diagram; a latency strip that fills in proportion to each stage's measured duration; a Reranked/Fusion-only toggle; and a source inspector with a Masked/Indexed toggle. Concurrency Control: PyTorch's Metal backend routes all operations through one shared command stream, so all model inference is serialised behind a process-wide lock, verified by a regression test that runs 72 retrievals across 12 threads.

VII. METHODOLOGY FOR DEVELOPING PRISM

- Needs Assessment: The problem was framed around find-by-description requests that filename and keyword search cannot answer invoices, identity documents, receipts, policies, and recordings and around the requirement that no document or query may leave the host. Requirement Definition: Support for ten file formats, citations that resolve to a page or timestamp, interactive retrieval latency, grounded answers with validated citations, abstention when evidence is missing, and masking of sensitive identifiers before any language model sees them.
- Platform Design: An embedded storage design forced by the no-container constraint, two embedding spaces fused at query time, and an interface built to make retrieval evidence visible rather than hidden. Development: Implemented in Python with FastAPI, PyTorch (Metal backend), the Qdrant client, SQLite FTS5, PyMuPDF, PP-OCRv4 through ONNX Runtime, faster-whisper and MLX, in phases recorded in eleven phase reports [27]. Testing: A 52-query labelled benchmark written before any ablation and never edited afterwards; a concurrency regression test of 72 retrievals across 12 threads; a redaction test paragraph containing both a valid identifier and a look-alike invoice number; and hand verification of generated answers. Deployment: A single 16 GB Apple silicon laptop with all models cached locally. A cold-cache ingest took 824.1 s, almost entirely model downloads, against 28.4 s with a

warm cache. Updating and Maintenance: Per-file event-driven re-indexing means a changed file costs one job rather than a corpus rebuilds, and the per-stage telemetry and audit trail make regressions visible as they occur.

VIII. PRISM PROTOTYPE, ALGORITHM, AND PROGRAM LOGIC

Algorithm 1 summarises the query path that the prototype executes for every search. Lookup queries stop after grouping; questions continue to grounded generation.

Algorithm 1: Prism query pipeline

```
Input : query q        Output: ranked files F, optional answer A
 1  e      ← embed_cpu(q)                      // bge-small, batch size 1
 2  L_dense ← qdrant.search(text_index, e)
 3  L_bm25  ← fts5.bm25(q)                     // body 1.0, heading 2.0, filename 3.0
 4  L_clip  ← qdrant.search(image_index, clip_text(q))
 5  S ← RRF({L_dense, L_bm25, L_clip}, k = 60)
 6  W ← top-20 of S, with 4 slots reserved for the best image candidates
 7  for c in W: c.score ← cross_encoder(q, text(c) or surrogate(c))
 8  if max(c.score) < −10.5: order W by best single-retriever rank
 9  F ← group W by file, keeping the best chunk per file
10  if q is a question:
11      A ← llm(q, redacted_text(F)); strip citations not found in F
12  return F, A
```

Figure 9 shows answer mode. A question produces a grounded answer with clickable citations that resolve to a page or timestamp, above the ranked files. The latency strip is proportional to measure per-stage duration; in this example generation (2,190 ms) dwarfs the entire 91 ms retrieval path, and the chips show which retrievers matched each file.

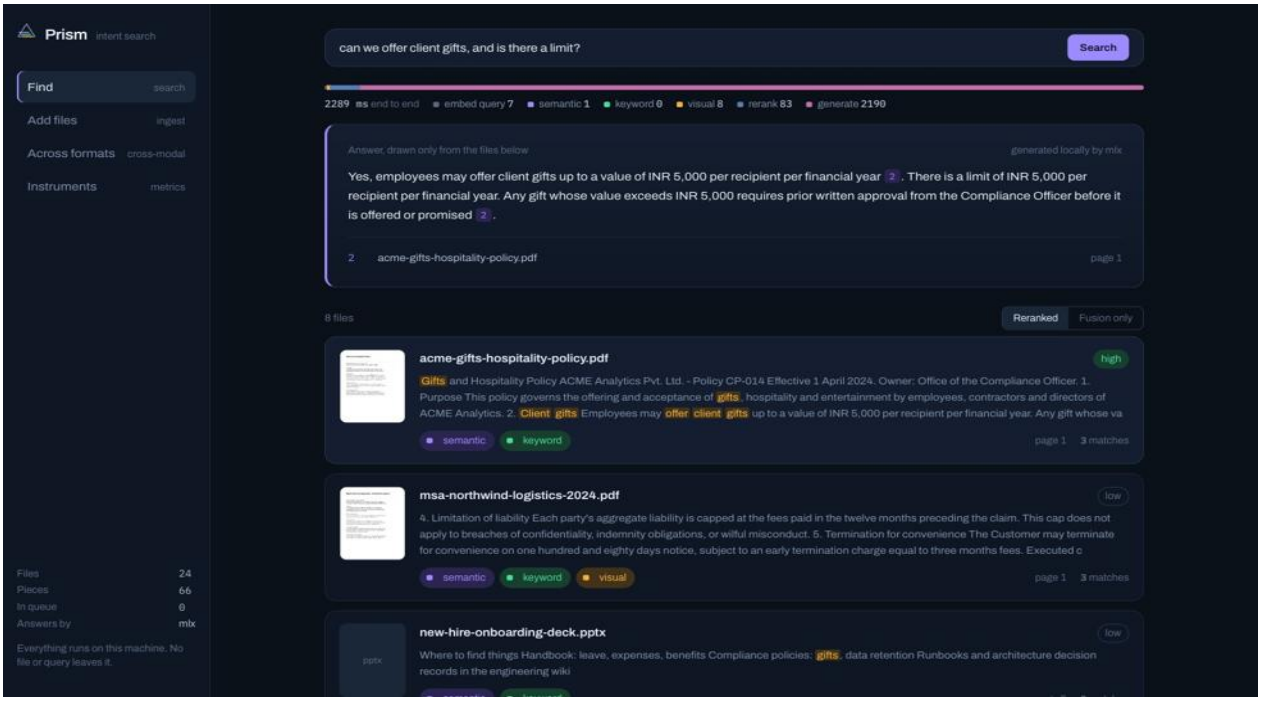


Figure 9: Answer Mode of Prism: Grounded Answer with Clickable Citations above the Ranked Files

Figure 10 shows cross-modal retrieval in both directions. On the left, a written description ranks images by CLIP similarity alone, placing the correct chart at 0.337 against 0.246 for the next candidate. On the right, the selected image becomes the query and returns its nearest neighbours, the two other charts at 0.504. No filename or caption is read.

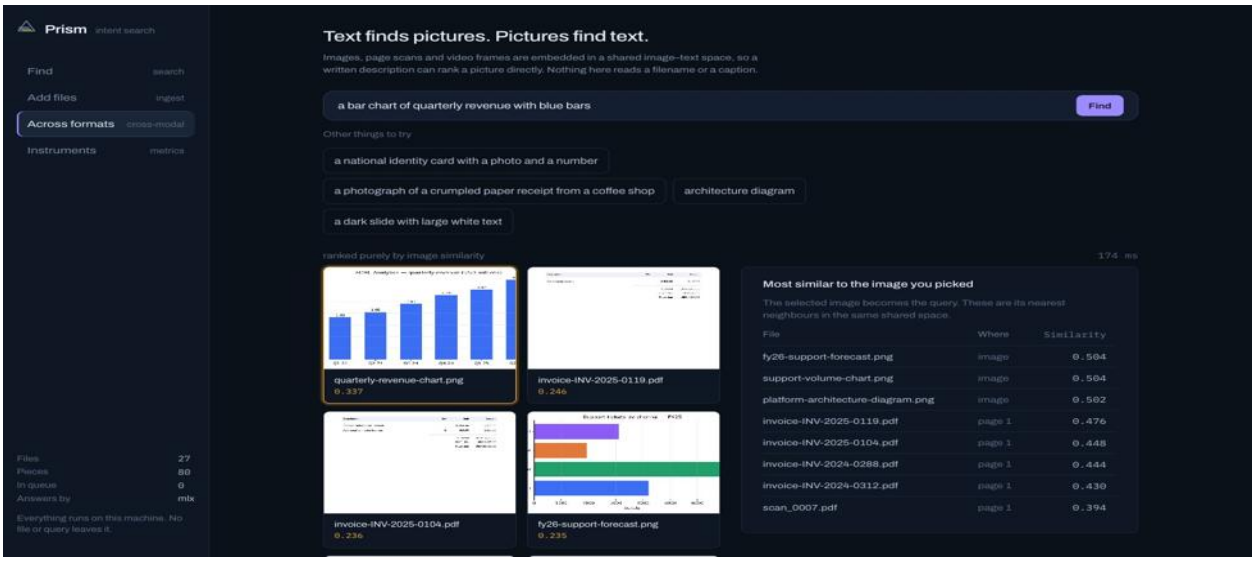


Figure 10: Cross-Modal Retrieval in Prism: Text-to-Image and Image-to-Image Search

Figure 11 makes the two-payload pattern visible. The identifier is masked as [AADHAAR] and the date of birth as [DOB] in the copy the language model receives, while the Masked/Indexed toggle reveals the unmasked text that remains in the retrieval index so the file stays findable by its number. The OCR artefact "GOVERNMENT OF EXAM PLIA" is a residual of the repair discussed in Section X.

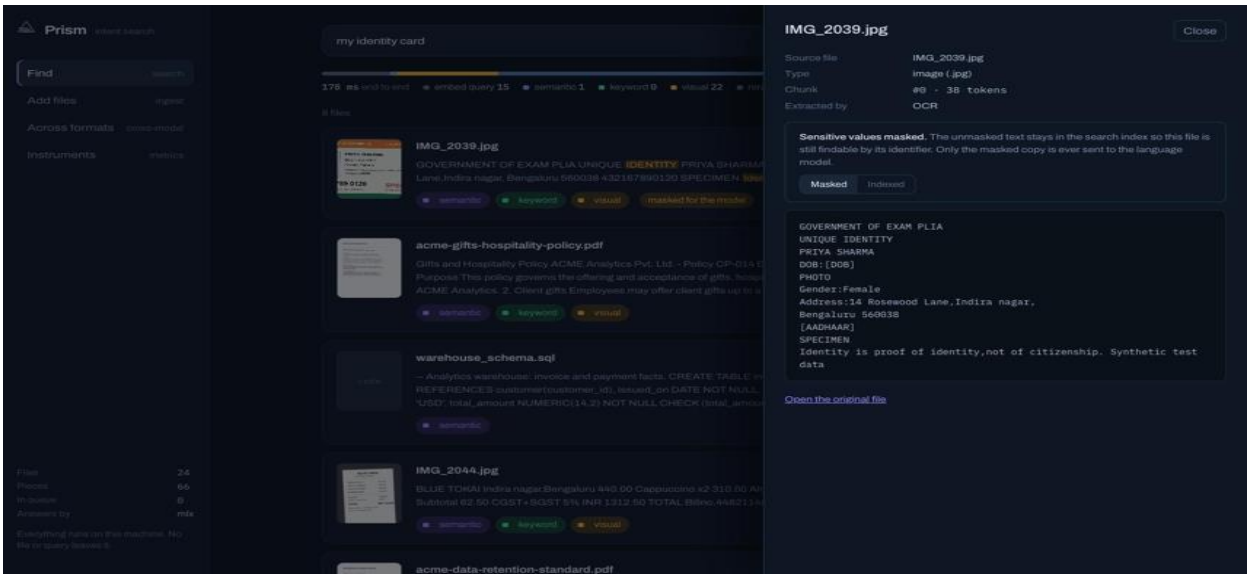


Figure 11: Two-Payload Redaction in Prism: Masked Copy for the LLM, Indexed Text for Retrieval

Figure 12 shows the Instruments page, which reads the same telemetry tables as the evaluation harness. It reports the live index 24 files, 66 retrievable pieces (44 text vectors and 22 image vectors) and 5 files carrying masked values and decomposes both query time and ingestion time by stage. Even in interactive use the shape matches Section IX: reranking dominates the query path and embedding dominates ingestion, followed by parsing and redaction, with index writes negligible. The absolute values are higher than in Table I because the panel averages all 69 queries logged in interactive browser sessions, including the first queries after the models load and answer-mode questions, whereas Table I reports warm, single-process harness runs.

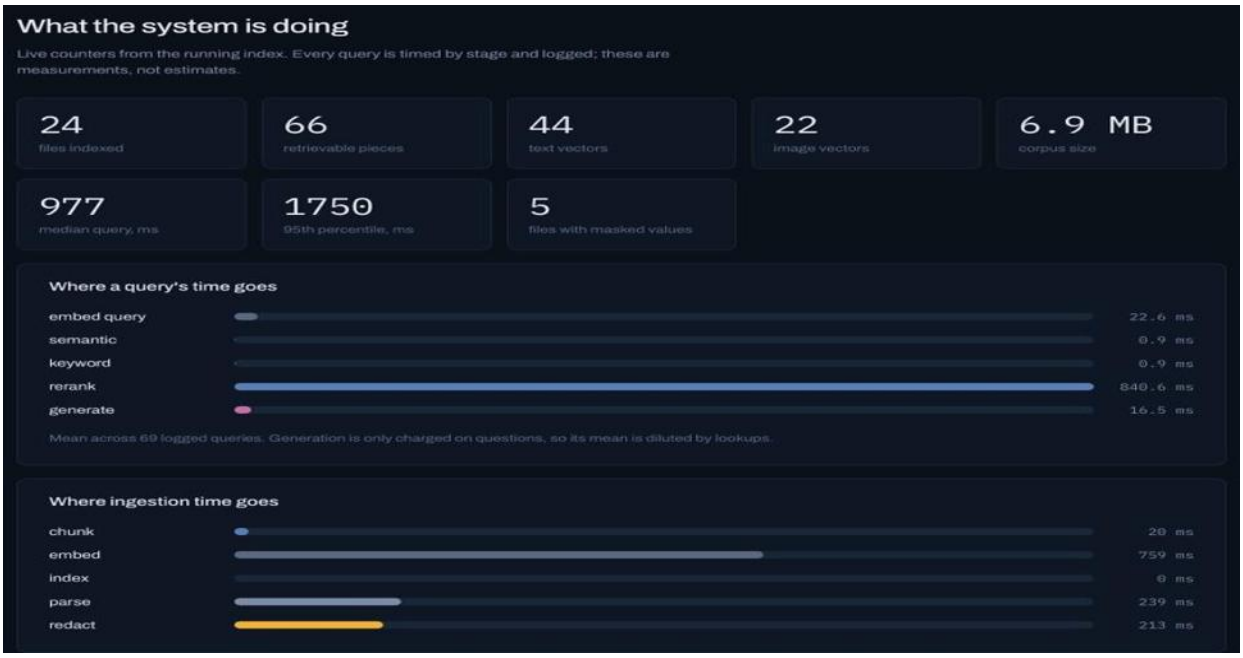


Figure 12: Instruments Page of Prism: Live Index Counters and Per-Stage Query and Ingestion Time

Figure 13 shows an intent lookup. The query "my identity card" shares nothing with the filename IMG_2039.jpg, yet the specimen card is ranked first with all three retrievers matching it, and it carries a "masked for the model" chip because its identifier and date of birth are redacted in the copy sent to generation while the indexed text keeps them searchable (Section XI). The latency strip shows the whole lookup at 628 ms, almost all of it query embedding (204 ms) and reranking (405 ms). The fourth result, a photographed café receipt, is matched lexically on "Paid by card ending 4242" a plausible-but-wrong file that the reranker correctly places below the intended one.

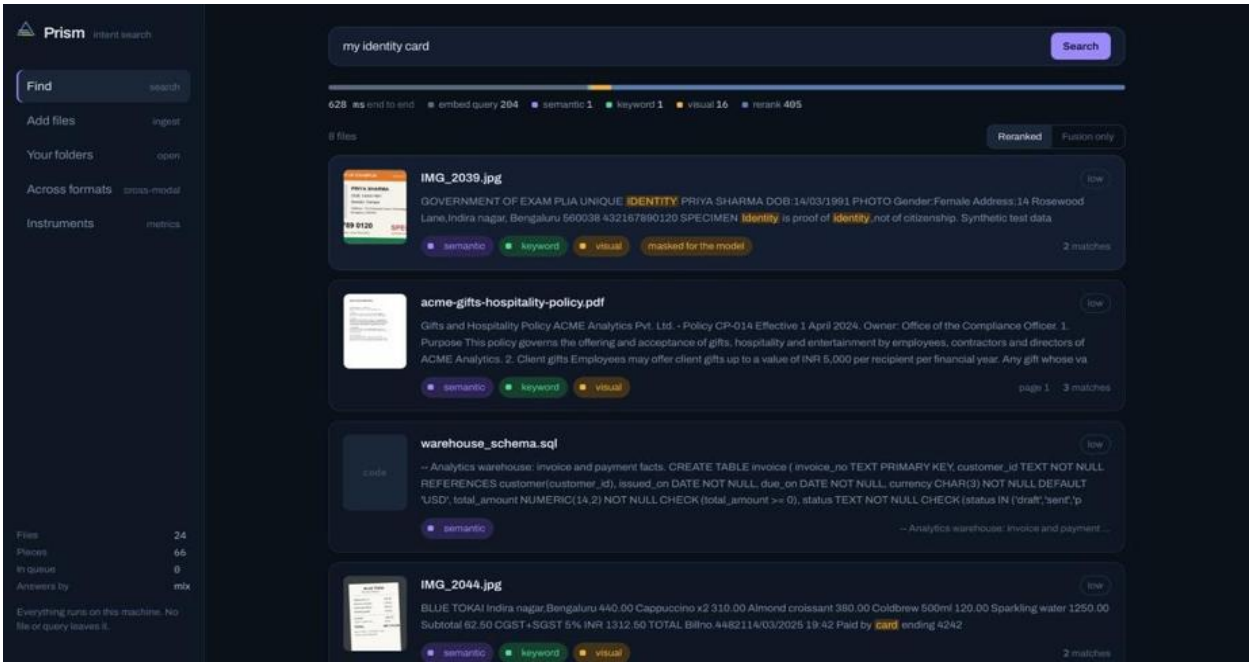


Figure 13: Intent-Based Lookup in Prism with Retriever Evidence Chips and the Per-Stage Latency Strip

Figure 14 shows the query log. Every search is recorded with its kind (lookup or answer), the number of files returned and its end-to-end time, and the same table feeds the counters in Figure 12. The answer-mode question "How long do we keep support call recordings?" took 1,310 ms, consistent with generation dominating answer latency, while repeated lookups such as "client gifts" complete in 172–183 ms. The earliest entries ("phishing training" at 1,603 ms and 1,466 ms) are the slowest, reflecting model warm-up. The log also captures realistic user phrasing the misspelling "adhar" and the vague "government id" which makes it a natural source of additional benchmark queries.



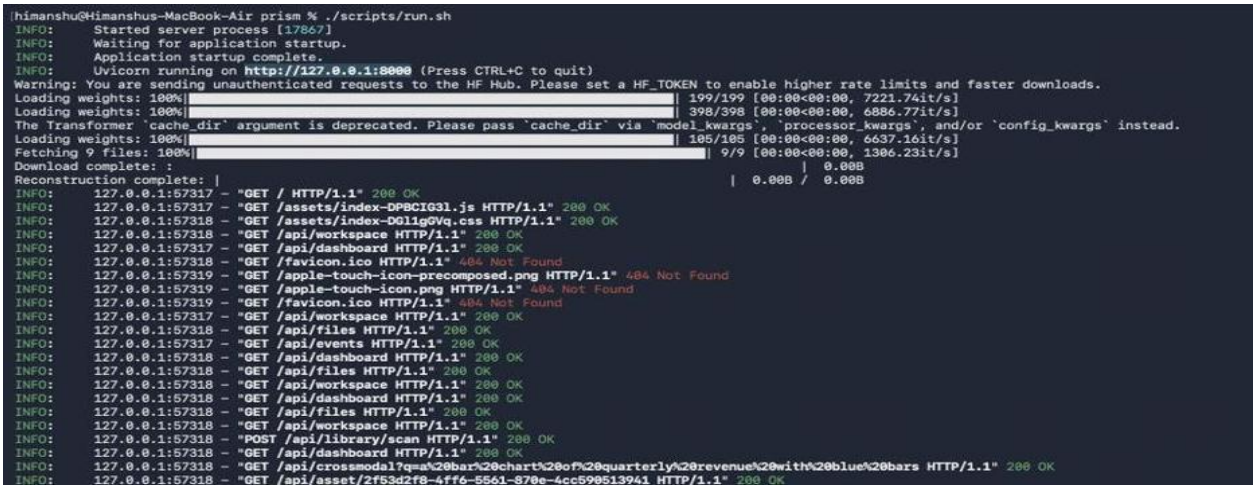
Recent queries

Query	Kind	Results	ms
my identity card	lookup	8	628
Indian	lookup	8	184
government id	lookup	8	220
adhar	lookup	8	358
How long do we keep support call recordings?	answer	8	1310
my identity card	lookup	2	233
client gifts	lookup	2	183
architecture diagram	lookup	3	163
my identity card	lookup	3	197
client gifts	lookup	3	172
phishing training	lookup	5	1603
phishing training	lookup	5	1466

Figure 14: Query Log of Prism Showing Query Kind, Result Count and End-to-End Latency

Figure 15 shows the system starting from its launch script. The application is served by Uvicorn on the loopback address 127.0.0.1:8000, the model weights are loaded from the local cache, and the log lines that follow are the interface's own API calls: workspace and dashboard polling, a library scan, and the cross-modal query "a bar chart of quarterly revenue with blue bars" from Figure 10 followed by the thumbnail fetches for its results. Every request in the log is addressed to the loopback interface. The startup log also records one deviation from the no-external-service constraint: the Hugging Face client contacts the model hub at startup to verify the cached weights (the "unauthenticated requests" warning, with 0.00 B downloaded). No document or query is transmitted, but a strictly offline deployment should set HF_HUB_OFFLINE=1 so that model loading never touches the network

.



```
himanshu@Himanshus-MacBook-Air prism % ./scripts/run.sh
INFO:     Started server process [17867]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
Warning: You are sending unauthenticated requests to the HF Hub. Please set a HF_TOKEN to enable higher rate limits and faster downloads.
Loading weights: 100%| | 199/199 [00:00<00:00, 7221.74it/s]
Loading weights: 100%| | 398/398 [00:00<00:00, 6886.77it/s]
The Transformer `cache_dir` argument is deprecated. Please pass `cache_dir` via `model_kwargs`, `processor_kwargs`, and/or `config_kwargs` instead.
Loading weights: 100%| | 105/105 [00:00<00:00, 6637.16it/s]
Fetching 9 files: 100%| | 9/9 [00:00<00:00, 1306.23it/s]
Download complete: :                                                    |  0.00B
Reconstruction complete: |                                             |  0.00B /  0.00B
INFO:     127.0.0.1:57317 - "GET / HTTP/1.1" 200 OK
INFO:     127.0.0.1:57317 - "GET /assets/index-DPBCIG3l.js HTTP/1.1" 200 OK
INFO:     127.0.0.1:57318 - "GET /assets/index-D0l1gGVq.css HTTP/1.1" 200 OK
INFO:     127.0.0.1:57318 - "GET /api/workspace HTTP/1.1" 200 OK
INFO:     127.0.0.1:57317 - "GET /api/dashboard HTTP/1.1" 200 OK
INFO:     127.0.0.1:57318 - "GET /favicon.ico HTTP/1.1" 404 Not Found
INFO:     127.0.0.1:57319 - "GET /apple-touch-icon-precomposed.png HTTP/1.1" 404 Not Found
INFO:     127.0.0.1:57319 - "GET /apple-touch-icon.png HTTP/1.1" 404 Not Found
INFO:     127.0.0.1:57319 - "GET /favicon.ico HTTP/1.1" 404 Not Found
INFO:     127.0.0.1:57317 - "GET /api/workspace HTTP/1.1" 200 OK
INFO:     127.0.0.1:57318 - "GET /api/files HTTP/1.1" 200 OK
INFO:     127.0.0.1:57317 - "GET /api/events HTTP/1.1" 200 OK
INFO:     127.0.0.1:57318 - "GET /api/dashboard HTTP/1.1" 200 OK
INFO:     127.0.0.1:57318 - "GET /api/files HTTP/1.1" 200 OK
INFO:     127.0.0.1:57318 - "GET /api/workspace HTTP/1.1" 200 OK
INFO:     127.0.0.1:57318 - "GET /api/dashboard HTTP/1.1" 200 OK
INFO:     127.0.0.1:57318 - "GET /api/files HTTP/1.1" 200 OK
INFO:     127.0.0.1:57318 - "GET /api/workspace HTTP/1.1" 200 OK
INFO:     127.0.0.1:57318 - "POST /api/library/scan HTTP/1.1" 200 OK
INFO:     127.0.0.1:57318 - "GET /api/dashboard HTTP/1.1" 200 OK
INFO:     127.0.0.1:57318 - "GET /api/crossmodal?q=a%20bar%20chart%20of%20quarterly%20revenue%20with%20blue%20bars HTTP/1.1" 200 OK
INFO:     127.0.0.1:57318 - "GET /api/asset/2f53d2f8-4ff6-5561-870e-4cc590513941 HTTP/1.1" 200 OK
```

Figure 15: Local Server Startup and API Request Log of Prism

IX. RESULTS AND EVALUATION

9.1 Experimental Setup

The corpus is 24 synthetic but realistic enterprise files totaling 7.3 MB across ten formats: native and scanned PDFs, DOCX, PPTX, XLSX, CSV, PNG, JPG, MP3, MP4 and source code. It includes four invoices, four policy documents, a contract, an incident postmortem, a specimen identity card photograph, a photographed receipt, two charts, an architecture
diagram, a narrated training recording and a narrated screen-capture video. The benchmark is 52 queries in ten categories
intent lookup, question and paraphrase (each also with an OCR-dependent variant), exact identifier, ASR-dependent, cross-modal and code each labelled with the file or files a competent human would accept. The queries and labels were written before any ablation was run and were not edited afterwards. Metrics are computed at file level because a file is what the system returns and what the labels name. All configurations run against the identical index, ablating one retriever arm at a time rather than reindexing.

9.2 Component Ablation

Table I and Figure 16 give the results. The full pipeline reaches P@1 0.962, R@5 1.000, MRR 0.981 and nDCG@10 0.984 at a median latency of 91.5 ms (p95 96.2 ms).

Table I: Retrieval Quality and Latency across Six Configurations (52 Labelled Queries, Identical Index)

Configuration	P@1	R@5	MRR	nDCG@10	p50 (ms)
Sparse only (BM25)	0.885	0.952	0.920	0.931	0.3
Dense only (bge-small)	0.769	0.933	0.848	0.882	5.7
Image only (CLIP)	0.269	0.462	0.365	0.429	6.8
Hybrid RRF	0.788	0.990	0.881	0.913	5.5
+ cross-modal	0.788	0.990	0.881	0.913	13.2
+ rerank (full pipeline)	0.962	1.000	0.981	0.984	91.5

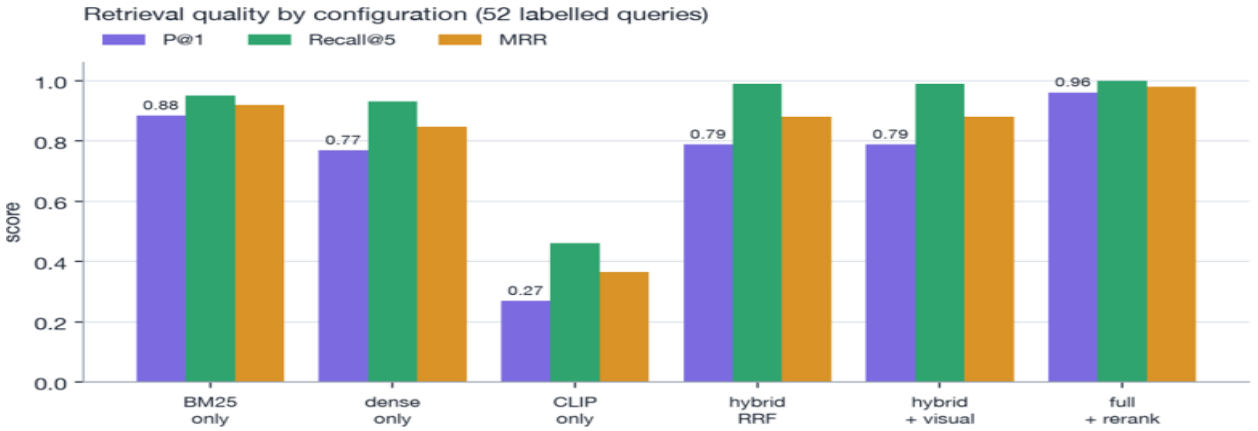


Figure 16: Retrieval Quality of Prism by Configuration (52 Labelled Queries)

Two findings are more interesting than the headline. Hybrid fusion reduced top-1 precision. BM25 alone achieves P@1 0.885. Adding the dense retriever lowers it to 0.788 while raising Recall@5 from 0.952 to 0.990. RRF is behaving exactly as designed anything either retriever liked is now present and the price is that a confident lexical first result is averaged against a mediocre dense one. For a user who reads only the first result, hybrid retrieval on this corpus was a worse system than BM25.

The reranker is what makes fusion pay. The cross-encoder converts the recall that fusion bought back into precision: P@1 rises from 0.788 to 0.962, MRR from 0.881 to 0.981 and R@3 from 0.962 to 1.000. Fusion and reranking are a pair: fusion without reranking is a regression, and reranking without fusion has a smaller candidate pool to work from.

9.3 Latency

Figure 17 decomposes the query path: query embedding 5.13 ms, dense search 0.55 ms, BM25 0.18 ms, CLIP search 6.96 ms and reranking 76.44 ms. The two index lookups together take under 1 ms, and reranking is 85.5% of the budget the largest quality gain in the system bought with the overwhelming majority of its time. The interface exposes this directly as a Reranked/Fusion-only toggle rather than hiding it. Generation, charged only on questions, adds a further 1.0–2.8 s and dominates end-to-end time for that query class. Figure 18 places the six configurations on a quality/latency plane: every configuration below 20 ms sits between MRR 0.37 and 0.92, and the only move above that band costs an order of magnitude in time.

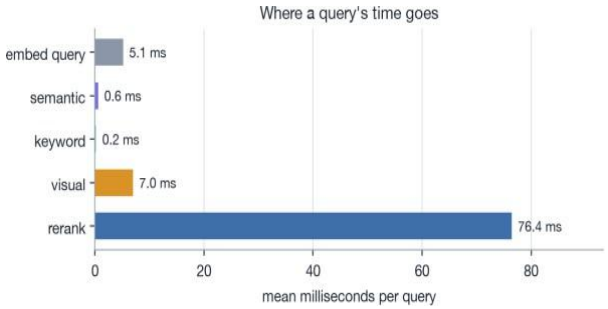


Figure 17: Mean Per-Stage Query Latency of the Full Pipeline

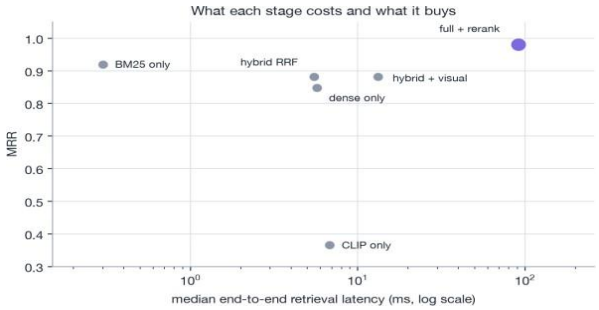


Figure 18: MRR against Median Retrieval Latency (Log Scale)

9.4 Ingestion Throughput

With a warm model cache, the 24-file corpus ingests in 28.4 s (0.85 files/s; median 91.8 ms per file). Stage totals across the corpus are: embedding 17,383 ms, parsing 5,475 ms, redaction 5,094 ms, chunking 410 ms, and index writes 3 ms. OCR and ASR paths cost 20–100× a native text extraction (Figure 19). The identical run on a cold cache took 824.1 s 29× slower almost entirely model downloads, which is why all throughput figures here are warm-cache and stated as such.

9.5 Per-Category Performance and Failure Analysis

Figure 20 breaks P@1 down by query category. Cross-modal is the only category below 0.92, and it is the category in which a single retriever, used alone, is perfect. Only two of 52 queries are wrong at rank 1. “Coffee shop receipt” returns the travel policy first and the receipt photograph second, although CLIP alone ranks the photograph first the dilution effect analysed in Section X. “Which invoices are overdue” returns a SQL schema file first and the intended spreadsheet second; the SQL file literally defines a view named overdue_invoice, so the label is arguably wrong rather than the system. It is recorded unchanged, because retrofitting labels to match system output is how an evaluation stops meaning anything.

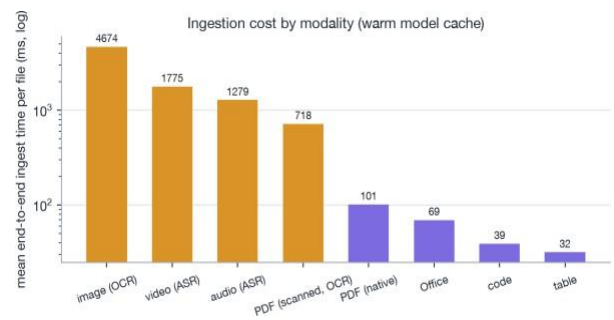


Figure 19: Ingest Cost per File by Modality (Warm Cache, Log Scale)

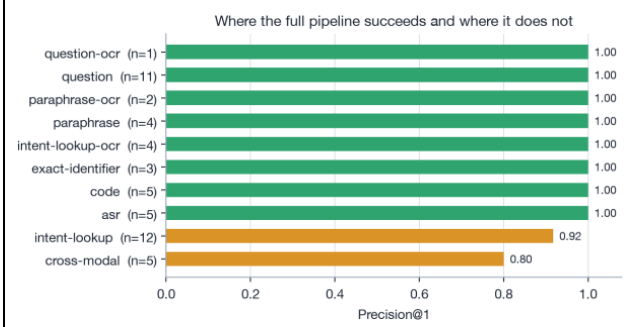


Figure 20: Precision@1 by Query Category for the Full Pipeline

X. DISCUSSION

10.1 Confident OCR that Lexical Retrieval Cannot Use

Running PP-OCRv4 on the specimen identity card returned essentially perfect characters at 0.98–0.99 reported confidence
and with every inter-word space removed: GOVERNMENTOFEXAMPLIA, PRIYASHARMA, Bengaluru560038. The retrieval consequence is asymmetric. Dense retrieval degrades but survives, because subword tokenization of GOVERNMENTOFEXAMPLIA still overlaps the tokens of "government identity". Lexical retrieval fails completely: FTS5 tokenizes PRIYASHARMA as one term that no user query can match. Before repair, BM25 did not return the identity card in its top three for "priya Sharma identity card" or for "government of example unique identity". The wider point is that OCR confidence measures character recognition, not retrievability; an ingestion pipeline gating on confidence a common design would have accepted this output silently. A conservative repair was added: camelCase splitting, digit/letter boundary splitting where the alphabetic run is at least four characters, and dictionary segmentation of remaining runs of nine or more characters, with an acceptance test requiring at least 60% of characters to land in pieces of four or more. Out-of-vocabulary fragments are re-glued so an unknown proper noun survives as one token rather than three meaningless ones. After repair, both queries return the card at rank 1
(BM25 scores 14.25 and 13.76). Residual errors remain and are recognition rather than spacing failures HIGH read as "Hl GH" which no post-processing recovers.

10.2 Cross-Modal Retrieval Contributed Less than the Demonstration Suggests

Table I shows the cross-modal arm changing no aggregate metric while adding 7.7 ms. Isolating the five cross-modal queries explains why: CLIP changes the top result on one of five. The other four are already solved by text retrieval, because those image files carry OCR-recoverable content chart axis labels, titles, legends that BM25 and the dense encoder match directly. On a corpus whose images contain readable text, vision-language embedding is largely redundant with OCR plus lexical search; it earns its place on images with no extractable text, and on the one query here where it was decisive.

10.3 CLIP's Usable Signal Is Visual Specificity, Not Query Length

Measured against the image index alone, "bar chart" leaves two chart images tied at 0.224 an arbitrary ordering while "a bar chart of quarterly revenue with blue bars" separates the correct one by 0.091. But "an architecture diagram with boxes and arrows" performs worse than "architecture diagram". Detail describing how a thing looks helps; detail describing what it means does not. CLIP's scores are also compressed into a narrow band with near-ties common at two decimal places, which independently confirms the decision to fuse on rank rather than score: at that resolution the magnitudes are untrustworthy while the ordering still carries information.

10.4 Fusion Improves the Average and Degrades the Best Case

On the five cross-modal queries, CLIP alone achieves P@1 1.00; the full pipeline, better on every other category and better overall, achieves 0.80 (Figure 20). Averaging a confident minority opinion with two indifferent ones loses information, and the best-single-retriever fallback recovers part of this but not all of it. This is evidence that a single fused ranking is the wrong shape for a genuinely multi-modal corpus, and that the missing component is a query router that decides which retrievers a query should consult before fusing the multi-agent decomposition that was not built.

10.5 A Single-Vector Page-Image Baseline

Every PDF page is rendered and embedded as one pooled CLIP vector, allowing retrieval with no OCR at all. On the seven OCR-dependent queries, this baseline reaches P@1 0.71 and Recall@5 1.00 it places the correct document in the top five every time against 1.00/1.00 for repaired OCR with BM25. This is not ColPali: late interaction [4], [16] is the mechanism that the paper contributes, and it was not implemented. But the strength of the pooled-vector floor is notable, and the comparison flatters OCR only because of the repair in Section 10.1; the unrepaired pipeline would have lost outright.

10.6 An Engineering Failure Worth Reporting

The API process aborted with SIGABRT and no Python traceback whenever a browser loaded the interface, while working correctly when driven from

Python. Two plausible hypotheses shell process reaping, then concurrent Metal initialization by two frameworks were wrong, and a lock added on the second theory changed nothing. The operating-system crash report identified the fault: seven threads inside PyTorch Metal kernels simultaneously, with the faulting thread aborting inside at: mps:MPS Stream:command Encoder () on a driver assertion. PyTorch's MPS backend uses one shared command stream that is not safe for concurrent use, and FastAPI executes synchronous endpoints on a thread pool, so ordinary concurrent requests produced the condition. The lesson generalizes: a SIGABRT with no Python traceback means the fault is below the interpreter, and the OS crash report is then the only artefact containing the answer. Reading it first would have cost less than either hypothesis.

XI. COMPLIANCE, ETHICS AND LIMITATIONS

11.1 Two Payloads

Masking text before embedding is known to distort the vector space: documents sharing sentinel tokens become mutual nearest neighbors, and recall collapses on exactly the sensitive material that most needs to be findable. Prism therefore keeps two payloads per chunk. The raw text stays in the retrieval indexes, so a user can still find their own identity card by typing its number. A redacted copy is stored alongside it and is the only text placed in a generation prompt; the context builder reads the redacted field and has no fallback to the raw one, because such a fallback is precisely how a redaction pipeline silently stops working. The interface makes this legible (Figure 11): the source inspector shows the masked copy by default, with a toggle to the indexed text and a one-sentence explanation of why both exist.

11.2 Checksums Supply the Precision

A bare twelve-digit pattern matches invoice numbers, order identifiers and tracking codes; on this corpus it produced more false positives than true ones. Aadhaar-format identifiers carry a Verhoeff check digit [21] and payment cards a Luhn check digit [22], so a candidate failing its checksum is discarded before masking. In a test paragraph containing a Verhoeff-valid identifier and the twelve-digit string 998877665544 labelled "Invoice no.", the identifier was masked and the invoice number was not. Transformer NER runs only when a checksum detector has already fired or a sensitivity keyword is present, bounding what is otherwise the most expensive stage of ingestion.

11.3 Limitations

Recall is unmeasured. There is no labelled PII ground truth, so precision is demonstrated and recall is not. A pattern-plus-checksum design is by construction blind to identifier formats it has no pattern for. Names are the weak point. Contextual patterns plus gated NER will miss a name appearing without a cue word in a chunk that triggered no other detector. Original files are stored unmodified, as they must be for the user to open them. The protection boundary is therefore "no sensitive value reaches the language model", not "the system holds no personal data". A deployment subject to GDPR or India's DPDP Act would need encryption at rest, access control and an erasure path that reaches the blob store none of which is implemented here. Masking is irreversible and carries no key; there is no re-identification path for an authorised caller.

11.4 Test Data

All corpus content is fabricated. The specimen identity card names a fictional country, is stamped "specimen" and "synthetic test data", and carries a checksum-valid number generated for this work. It exists so the redaction path can be tested end to end; no real personal data was used at any point.

11.5 Threats to Validity

The headline figures are high enough to mislead, so their limits are stated directly. Corpus scale. Twenty-four seeded files; P@1 0.962 means 50 of 52 queries. At this size, the dense retriever's top-40 is essentially the entire corpus, so RRF distributes shallow credit almost everywhere and the reranker chooses from a candidate set that nearly always contains the answer. None of these numbers should be expected to hold at 10^5 documents; BM25 would be expected to degrade fastest and the ranker's contribution to grow. Corpus provenance. The documents and the queries were authored for this work, so vocabulary overlap is more favorable than in a real deployment despite deliberate paraphrase categories. Single-user latency. All timings are single-process and warm-cache, with the inference lock uncontended. Concurrent load serializes on that lock by design. Generation is spot-checked, not scored. Four question/answer pairs were verified by

hand against sources, including one correct abstention. There is no faithfulness metric of the kind RAGAS [26] provides. Parameters are argued, not swept. The RRF constant k = 60 [3], the reserved-slot count, the reran confidence floor, and the scanned-page threshold are justified by reasoning and single observations rather than by parameter sweeps.

XII. CONTRIBUTION AND FINDINGS

The Prism project provides a private, locally-runnable and evidence-first search and question-answering system for enterprise knowledge that is scattered across documents, images, recordings and code. By combining event-driven per-file ingestion, three complementary retrievers, rank fusion, cross-encoder reranking, citation-validated local generation and checksum-based redaction, Prism lets a user find a file by describing it and trace every answer to a page or a timestamp, without any document leaving the machine. The measurements were more useful than the system. Three expectations did not survive contact with them: that hybrid fusion improves retrieval (it lowered top-1 precision, and only reranking rescued it); that a vision-language retriever carries multi-modal search (it changed one result in five, because OCR had already made those images findable); and that OCR confidence indicates whether text is usable (0.99-confidence output was unusable by lexical search). Each was found only by ablating a component that a purely aggregate evaluation would have reported as harmless.

XIII. CONCLUSION AND FUTURE WORK

Prism demonstrates that a multi-modal, citation-grounded retrieval system spanning ten formats can run entirely on one consumer laptop, with sub-100 ms median retrieval and locally generated grounded answers, under constraints no container runtime, no external service that eliminate most conventional architectural choices. The full pipeline reaches Precision@1 of 0.962 and Recall@5 of 1.000 on a 52-query labelled benchmark at a median retrieval latency of 91.5 ms, while the component ablation shows which parts of that result each component actually earned. The highest-value next steps follow directly from the failures. First, a query router that decides which retrievers a query should consult before fusing, since fusion currently costs 0.20 P@1 on the queries where a single retriever is perfect.

Second, late interaction over the page renders that already exist, to separate the contribution of looking at a page from that of multi-vector scoring. Third, scale: repeat the ablation two orders of magnitude larger, where the component ordering reported here is expected to change. Fourth, measured faithfulness for generated answers, and a labelled PII set so that redaction recall can be reported rather than omitted. Reproducibility. The system, the 52-query labelled benchmark, the evaluation harness, the chart generators and eleven phase reports documenting each decision and failure as it occurred are provided as accompanying artefacts [27]. Every figure and every number in this paper is produced by the running system or by a script over data it emitted; none was entered by hand.

REFERENCES

[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-T. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[2] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-T. Yih, "Dense passage retrieval for open-domain question answering," in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6769–6781.

[3] G. V. Cormack, C. L. A. Clarke, and S. Buettcher, "Reciprocal rank fusion outperforms Condorcet and individual rank learning methods," in *Proc. 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2009, pp. 758–759.

[4] M. Faysse, H. Sibille, T. Wu, B. Omrani, G. Viaud, C. Hudelot, and P. Colombo, "ColPali: Efficient document retrieval with vision language models," *arXiv preprint arXiv:2407.01449*, 2024.

[5] G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open domain question answering," in *Proc. 16th Conference of the European Chapter of the*

Association for Computational Linguistics (EACL), 2021, pp. 874–880.

[6] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2023.

[7] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks," in *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019, pp. 3982–3992.

[8] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[9] T. Formal, B. Piwowarski, and S. Clinchant, "SPLADE: Sparse lexical and expansion model for first stage ranking," in *Proc. 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 2288–2292.

[10] R. Nogueira and K. Cho, "Passage re-ranking with BERT," *arXiv preprint arXiv:1901.04085*, 2019.

[11] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, and F. Wei, "Text embeddings by weakly-supervised contrastive pre-training," *arXiv preprint arXiv:2212.03533*, 2022.

[12] S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, D. Lian, and J.-Y. Nie, "C-Pack: Packed resources for general Chinese embeddings," in *Proc. 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024, pp. 641–649.

[13] Radford, J. W. Kim, C. Hallacy, R. Ramesh, A. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proc. 38th International Conference on Machine Learning (ICML)*, 2021, pp. 8748–8763.

[14] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 824–836, 2020.

[15] Qdrant Team, "Qdrant: Vector search engine," software, Apache-2.0 license, 2026. [Online]. Available: https://github.com/qdrant/qdrant. [Accessed: Sep. 9, 2026].

[16] O. Khattab and M. Zaharia, "ColBERT: Efficient and effective passage search via contextualized late interaction over BERT," in *Proc. 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 39–48.

[17] Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *Proc. 40th International Conference on Machine Learning (ICML)*, 2023, pp. 28492–28518.

[18] G. Klein, "faster-whisper: Fast inference for Whisper using CTranslate2," software, 2026. [Online]. Available: https://github.com/SYSTRAN/faster-whisper. [Accessed: Sep. 9, 2026].

[19] Y. Du, C. Li, R. Guo, X. Yin, W. Liu, J. Zhou, Y. Bai, Z. Yu, Y. Yang, Q. Dang, and H. Wang, "PP-OCR: A practical ultra lightweight OCR system," *arXiv preprint arXiv:2009.09941*, 2020.

[20] Prism Project, "Multimodal RAG for enterprise knowledge: Architecture, models, retrieval and deployment," internal technical report accompanying this work, 2026.

[21] J. Verhoeff, *Error Detecting Decimal Codes*, Mathematical Centre Tracts, no. 29. Amsterdam, The Netherlands: Mathematisch Centrum, 1969.

[22] International Organization for Standardization, *ISO/IEC 7812-1: Identification Cards—Identification of Issuers—Part 1: Numbering System*, 2017.

[23] SQLite Consortium, "SQLite FTS5 extension," 2026. [Online]. Available: https://www.sqlite.org/fts5.html. [Accessed: Sep. 9, 2026].

[24] Qwen Team, "Qwen2.5 technical report," *arXiv preprint arXiv:2412.15115*, 2024.

[25] A. Hannun, J. Digani, A. Katharopoulos, and R. Collobert, "MLX: Efficient and flexible machine learning on Apple silicon," software,

2023. [Online]. Available: https://github.com/ml-explore/mlx.

[26] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, "RAGAS: Automated evaluation of retrieval augmented generation," in *Proc. 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, 2024, pp. 150–158.

[27] Prism Project, "Prism build log: Phase reports 00–10," accompanying artefact, reports/00-index.md, 2026.

[28] G. D. K., M. K. Singh, and M. Jayanthi, Eds., *Network Security Attacks and Countermeasures*. Hershey, PA, USA: IGI Global, 2016, doi: 10.4018/978-1-4666-8761-5.

[29] R. M. Balajee, M. K. Jayanthi Kannan, and V. Murali Mohan, "Image-based authentication security improvement by randomized selection approach," in *Inventive Computation and Information Technologies*. Singapore: Springer, 2022, pp. 61–71.

[30] M. K. J. Kannan, N. Aaijaz, K. G. Mani, and V. Tewari, *The Future of Innovation and Technology in Education: Trends and Opportunities*. S&M Publications, 2025.

[31] B. R. M. and J. K. M. Kannan, "Intrusion detection on AWS cloud through hybrid deep learning algorithm," *Electronics*, vol. 12, Art. no. 1423, 2023, doi: 10.3390/electronics12061423.

[32] K. L. S. Kumar and M. K. J. Kannan, "A survey on driver monitoring system using computer vision techniques," in *Innovative Computing and Communications*, A. E. Hassanien, S. Anand, A. Jaiswal, and P. Kumar, Eds., Lecture Notes in Networks and Systems, vol. 1021. Singapore: Springer, 2024, doi: 10.1007/978-981-97-3591-4_21.

[33] M. K. J. Kannan, "A bird's eye view of cybercrimes and free and open-source software's to detoxify cybercrime attacks—an end user perspective," in *Proc. 2nd International Conference on Anti-Cyber Crimes (ICACC)*, Abha, Saudi Arabia, 2017, pp. 232–237, doi: 10.1109/Anti-Cybercrime.2017.7905297.

[34] M. K. J. Kannan, S. Kumar, P. T. Kalaivaani, and G. Tripathi, *Artificial Intelligence and Blockchain Technology for Human Resource Management*. Scientific International Publishing House, 2025.