

Volume 2, Issue 2

Research Article

Date of Submission: 21 August, 2026

Date of Acceptance: 17 September, 2026

Date of Publication: 24 September, 2026

Scholar Introduction to the Educational Knowledge Fragmentation Crisis

Ayushmaan Singh Bundela* 

Independent researcher, India

***Corresponding Author:**

Ayushmaan Singh Bundela, Independent researcher, India.

Citation: Bundela, A. S. (2026). SCHOLAR Introduction to the Educational Knowledge Fragmentation Crisis. *Adv Brain-Computer Interfaces Neural Integr*, 2(2), 01-12.

In modern technical and engineering pedagogical environments, undergraduate students are subjected to intense cognitive demands, particularly concentrated within brief mid-semester and end-semester evaluation cycles [1]. A primary structural deficiency contributing to academic performance degradation, cognitive overload, and elevated anxiety is the profound reliance on highly fragmented, unstructured learning materials. Within the traditional university ecosystem, students aggregate information from disparate and disconnected avenues: unstructured lecture audio, incomplete or low-resolution presentation slide sheets distributed across disjointed messaging platforms, and highly heterogeneous, overlapping handwritten peer notes [1].

Standard digital educational infrastructures, such as enterprise Learning Management Systems (LMS) like Canvas, Blackboard, or Moodle, or consumer cloud storage solutions like Google Drive and Microsoft OneDrive, function strictly as static, passive data repositories [1]. These systems provide secure file hosting and role-based access but possess absolutely no content-level comprehension or orchestration capabilities. They cannot read, contextualize, deduplicate, or synthesize the binary assets they store [1]. Consequently, students squander critical cognitive resources and study hours on administrative file organization and manual cross-referencing rather than deep conceptual mastery.

When multiple students upload their individual lecture notes for a specific syllabus module, the resulting repository is riddled with informational redundancies, contradictory formatting, and structural gaps [1]. Manually comparing these distinct vectors of the exact same lecture content is an unscalable burden. To resolve this friction, "SCHOLAR" (developed under the internal project moniker "AlgoQuest") was engineered as a centralized, collaborative platform that functions as an advanced orchestration engine for academic synthesis [1]. SCHOLAR leverages state-of-the-art Natural Language Processing (NLP), Large Language Models (LLMs), and Retrieval-Augmented Generation (RAG) to autonomously ingest heterogeneous documents, execute high-dimensional semantic deduplication, and compile an authoritative, syllabus-aligned "Perfect Note," while simultaneously providing an interactive AI tutoring interface [1].

This report provides an exhaustive, peer-level technical dissection of the publishing status and architectural mechanisms underlying the SCHOLAR platform, bridging the theoretical developments in machine learning—such as Transformer architectures and LayoutLM spatial analysis—with practical deployment in an educational technology ecosystem [2,3].

Pedagogical Foundations of Computer-Supported Collaborative Learning (CSCL)

The transition from isolated learning paradigms to Computer-Supported Collaborative Learning (CSCL) has been heavily scrutinized in educational technology research. Collaborative note-taking is a critical instructional strategy that distributes the cognitive load of recording information across a group, allowing individual learners to focus on synthesis, active listening, and comprehension [4]. Traditionally, this is executed via rotational note-taking, where a different student acts as the designated scribe each session, or role-based note-taking, where students divide conceptual coverage [4].

However, human-driven collaborative note-taking often results in chaotic, centralized documents that require extensive manual editing to become viable study guides [5]. The integration of artificial intelligence into CSCL environments fundamentally alters this dynamic. AI agents can act as knowledgeable learning peers or centralized “director agents,” continuously monitoring the dialogue and shared documents to extract salient points, enforce pedagogical strategies, and ensure equitable knowledge distribution [6].

Through cluster analysis and epistemic network analysis of CSCL environments, researchers have identified distinct learner typologies, such as “active questioners” who engage in cognitive negotiation, and “responsive navigators” who guide collaborative flow [6]. Platforms like CollaClassroom have demonstrated that LLMs can act as personal and group assistants grounded on shared session documents, supporting fluid transitions between solo and group sensemaking [7].

SCHOLAR operationalizes these advanced pedagogical theories by removing the burden of manual document merging from the student cohort entirely. By allowing all students to asynchronously upload their individual interpretations and notes of a lecture, the system captures a wide diversity of cognitive perspectives [1]. The underlying AI pipeline then takes on the role of the director agent, synthesizing these perspectives into a singular, highly coherent educational asset that benefits the entire collective without requiring synchronous, real-time manual editing [6].

System Design and Architectural Paradigm

The SCHOLAR architecture is constructed upon a decoupled, high-performance three-tier software paradigm designed to handle both long-running, asynchronous AI generation jobs and low-latency interactive vector queries [1]. The structural design adheres to strict non-functional requirements (NFRs) ensuring scalability, reliability, and precision.

Architectural Layer	Selected Framework / Tool	Specific Version	Core Engineering Justification
Frontend Mobile UI	Kotlin / Jetpack Compose	Latest	Native state persistence for streaming text views and responsive component logic.
Application Gateway	FastAPI (Python)	v0.111	Asynchronous concurrency for long LLM inference waits and WebSocket routing.
AI Orchestration	LangChain Core / AICore	v0.2.0	Out-of-the-box Map-Reduce pipelines and native Android on-device inference via Gemini Nano.
Vector Storage	Pinecone & ObjectBox	Cloud/Native	Cloud vector indexing combined with on-device vector storage via ObjectBox for local retrieval.

Relational Database	PostgreSQL	v16.2	ACID-compliant storage for user metadata, credentials, and document logging.
1			

Native Android Interface Ecosystem

The client interface is built using Kotlin and Jetpack Compose, chosen for their declarative UI and component-driven architectures essential for rendering complex, state-heavy interactive mobile workspaces [1]. The user experience is anchored by a mobile-optimized control workspace: the primary view presents a scrollable, dynamically generated Markdown document viewer, while an interactive bottom-sheet or swipeable tab houses the interactive RAG conversational panel [1]. This streamlined mobile UI significantly reduces student onboarding friction, a critical operational feasibility factor [1].

A highly innovative UX feature is the “Contextual Highlight Trigger”—when a user highlights a complex technical phrase within the synthesized document, the text automatically populates the chat widget’s input field [1]. This lowers the interaction cost required to receive targeted explanations and mimics the organic process of asking a professor for clarification on a specific textbook paragraph.

Application Gateway and Routing

The middle tier utilizes FastAPI, an extremely fast asynchronous web framework built on Python, serving as the cloud backend for the Android client [1]. FastAPI is strictly required because traditional synchronous web servers block execution threads while waiting for upstream LLM inference APIs to return payloads. Given that multi-document summarization tasks can take tens of seconds, blocking threads would cripple concurrent capacity [1]. FastAPI leverages Python’s asyncio library to handle long LLM generation waits in the background while simultaneously managing active streaming token operations via WebSockets (consumed via Kotlin Coroutines and OkHttp on the client) for the live chat interface [1].

The system exposes explicit HTTP REST endpoints, including:

- POST /api/v1/auth/register: Secure login and account verification using institutional email credentials.
- POST /api/v1/materials/upload: A multipart form route that ingests binary files, initiates asynchronous text extraction, and logs metadata.
- GET /api/v1/synthesize/generate/{module_id}: Triggers the LangChain map-reduce deduplication process to output a compiled markdown document.
- POST /api/v1/chat/query: Receives user queries, triggers the vector database similarity search, and streams back RAG answers [1].

Persistence and State Management

Data persistence is deliberately split across optimized environments to handle fundamentally different data modalities. Relational data, including user account credentials, course cataloging, role-based access controls (RBAC), and document metadata, is managed by an ACID-compliant PostgreSQL database [1].

For high-dimensional spatial semantic indexing, SCHOLAR employs a hybrid approach. While cloud-native vector databases like Pinecone offer massive scalability across user cohorts, the mobile client also integrates ObjectBox, the first on-device vector database designed specifically for Java and Kotlin environments. By utilizing ObjectBox, the application can securely store specific document vectors directly on the Android device and execute sub-millisecond similarity searches utilizing the Hierarchical Navigable Small World (HNSW) algorithm entirely offline. This guarantees that data never has to leave the device for local queries, vastly improving privacy and reducing network latency.

Multi-Modal Document Ingestion and Spatial Text Extraction

The foundation of any knowledge synthesis engine is the fidelity of its initial data extraction. Because academic materials are intrinsically heterogeneous, SCHOLAR deploys a variable ingestion engine capable of parsing digital PDFs, presentation slides, plain text markdown, and raw images of handwritten notes [1].

Processing Heterogeneous Formats

Different file types necessitate specific extraction runtimes. Digital PDF handouts, which often contain multi-column layouts and embedded text vectors, are processed using robust libraries like PyPDF or PDFPlumber [1]. Classroom presentations, which encapsulate text blocks, bullet arrays, and vector figures, require specific parsing algorithms via Python-PPTX [1].

The most significant computational and qualitative challenge lies in processing handwritten notebook snapshots. These images often suffer from low contrast, variable skew, illumination gradients, and highly idiosyncratic cursive scripts [1]. SCHOLAR mitigates extraction degradation through pre-processing contrast-enhancement scripts prior to routing the image through Optical Character Recognition (OCR) engines like Tesseract (initially developed by Hewlett-Packard and currently maintained by Google) or commercial cloud-based vision APIs [1].

Document Layout Analysis via LayoutLM

Advanced document understanding relies not just on character extraction, but on holistic spatial awareness. Traditional OCR merely extracts strings of text from left to right, completely destroying the semantic hierarchy of tables, sidebars, and captions [8]. To resolve this, modern systems look to spatial modeling architectures like LayoutLM [3].

LayoutLM pre-trains on both text and layout information to deeply understand document imagery [9]. Inspired by the masked language models of BERT, LayoutLM utilizes a Masked Visual-Language Model (MVLN) approach. During pre-training on massive corpora like the IIT-CDIP Test Collection (which contains over 11 million scanned document images), the model randomly masks input tokens but retains their corresponding 2-D position embeddings (bounding box coordinates) [10]. The model is forced to predict the masked text using both the surrounding linguistic context and the spatial layout clues [10].

By bridging the gap between visual and language modalities, LayoutLM can execute semantic labeling—aggregating words as semantic entities (e.g., classifying a string of text as a ‘header’ versus a ‘footer’ versus ‘primary content’) [8]. Incorporating spatial heuristics ensures that when SCHOLAR extracts text from complex engineering schematics or multi-column research papers, the sequential flow of information is preserved perfectly before it is passed to the chunking algorithms.

Text Segmentation Strategy: Recursive Character Chunking

Following successful extraction, the raw, normalized text strings must be segmented. Foundation LLMs operate under strict context window constraints (maximum token limits per inference request).¹ Feeding a massive, concatenated string of five different 20-page lecture notes directly into an LLM will inevitably result in token overflow errors, API rejection, or severe context degradation (a phenomenon where the model “forgets” information located in the middle of a massive prompt) [1].

To circumvent this, SCHOLAR utilizes LangChain’s RecursiveCharacterTextSplitter algorithm [1].

This algorithm is fundamentally superior to rigid, character-count-based splitters (like a basic FixedTokenChunker) because it respects the semantic hierarchy and natural flow of human language [11].

The Mechanics of Recursive Splitting

A basic FixedTokenChunker simply counts tokens and slices the string arbitrarily. For example, setting a chunk size to 100 tokens will cut the text exactly at the 100th token, frequently slicing a sentence or even a complex word perfectly in half, destroying the contextual meaning [11]. Conversely, the recursive chunker operates by attempting to divide the text using a prioritized, hierarchical list of natural separators [11]. It initiates the split using large structural boundaries, typically double newlines (\n\n), which universally represent paragraph breaks in Markdown and standard text [11]. If the resulting chunk still exceeds the hard-coded token limit, the algorithm recursively falls back to the next separator in the hierarchy, such as single newlines (\n), followed by sentence-ending punctuation (e.g., periods, exclamation marks), spaces, and ultimately individual characters as a last resort [11].

This ensures that the text is broken into meaningful, logical units. In the SCHOLAR architecture, the chunk limits are strictly locked at 1,000 tokens [1].

The Importance of Sliding Window Overlap

When text is severed into discrete chunks, the contextual meaning at the exact boundary of the split is highly vulnerable. For example, a pronoun (“It”) at the very beginning of Chunk B might refer to a proper noun (“The backpropagation algorithm”) at the very end of Chunk A. To preserve semantic continuity across these artificial boundaries, SCHOLAR implements a 200-token sliding window overlap [1].

Consequently, the final 200 tokens of the preceding chunk are explicitly duplicated at the beginning of the subsequent chunk [11]. This overlapping buffer provides the downstream LLM with the necessary trailing context, ensuring that coreferences, acronym definitions, and continuous logical arguments are not broken during the vectorization or synthesis phases.

High-Dimensional Vector Embedding and Semantic Deduplication

Once the documents are ingested and recursively chunked, the system faces its primary algorithmic challenge: identifying and eliminating redundant information submitted by multiple students. When four students attend the exact same B.Tech lecture on Big Data Systems, their notes will cover identical core concepts but using vastly different vocabulary, phrasing, and syntax [1].

Traditional syntactic deduplication relies on exact string matching or fuzzy hashing, which completely fails when students paraphrase [12]. Semantic deduplication, however, identifies pairs of data that are semantically similar but not exactly identical lexically [13]. In the context of large-scale LLM pre-training, semantic deduplication techniques (such as the SemDeDup algorithm) have been shown to drastically reduce dataset size by up to 50% while maintaining or improving model performance by eliminating redundant memorization triggers and “mosaic memory” phenomena [13].

Sentence-BERT (SBERT) and Siamese Networks

To detect semantic overlap, the text chunks must be converted into a mathematical format that a machine can evaluate. SCHOLAR achieves this by transforming the raw text chunks into 1536-dimensional semantic coordinate vectors using advanced text-embedding models [1]. The underlying architecture for generating high-quality embeddings relies heavily on modifications of the BERT (Bidirectional Encoder Representations from Transformers) network [14]. BERT, published by Devlin et al. in 2019, revolutionized NLP by pre-training deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers [14].

However, standard BERT architectures are incredibly inefficient for large-scale clustering. To calculate the semantic similarity between two sentences, standard BERT requires both sentences to be fed into the network simultaneously (separated by a `` token) to calculate an attention-based similarity score [15]. Finding the most similar pair in a collection of just 10,000 sentences requires 50 million inference computations, which takes roughly 65 hours on a modern GPU [15].

To solve this, researchers introduced Sentence-BERT (SBERT) at EMNLP 2019 [15]. SBERT utilizes Siamese and triplet network structures to derive a fixed-size, semantically meaningful sentence embedding for each chunk independently [15]. It adds a pooling operation to the output of BERT to derive a fixed-sized vector [16]. Once these independent embeddings are generated, the similarity between any two text chunks can be calculated almost instantaneously (reducing the 65-hour computation down to ~5 seconds) using highly efficient mathematical distance metrics [15].

Cosine Similarity Mathematics and Spatial Indexing

The primary metric used to evaluate the distance between these high-dimensional vectors in the vector database is Cosine Similarity [17]. Cosine similarity measures the cosine of the angle between two non-zero vectors in an inner product space. It is particularly effective for NLP text analysis because it is independent of document length (magnitude) and focuses entirely on the orientation (the semantic “direction”) of the vectors [18].

Mathematically, the dot product similarity for two vectors **a** and **b** of *n* dimensions is expressed as the sum of the products of their corresponding components: [18]

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + a_3 b_3 + \cdots + a_n b_n$$

The cosine similarity score is then calculated by dividing this dot product by the product of the vectors’ Euclidean magnitudes:

$$\text{Cosine Similarity} = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|} = \frac{\sum_{i=1}^n a_i b_i}{\sqrt{\sum_{i=1}^n a_i^2} \sqrt{\sum_{i=1}^n b_i^2}}$$

By calculating these scores across all uploaded chunks, the system can dynamically cluster semantically identical paragraphs [12]. For instance, if Student A’s notes contain the string “The neural network adjusts its weights via backpropagation,” and Student B’s notes state “Errors are propagated backward to update the model parameters,” SBERT maps them to highly proximate locations in the 1536-dimensional space, yielding a high cosine similarity score (e.g., >0.85) [12]. SCHOLAR flags these highly proximate chunks as redundant and prepares the cluster for the Map-Reduce synthesis pipeline [1].

The AI Synthesis Engine: Map-Reduce Orchestration via LangChain

With the text chunked, embedded, and semantically clustered, the actual generation of the unified “Perfect Note” relies on LangChain’s orchestration of foundation LLMs utilizing a Map-Reduce algorithmic paradigm [1].

The Map-Reduce method is specifically designed to summarize massive datasets or large documents that far exceed a single language model’s token limit [19]. The LangChain framework provides composable chains (MapReduceDocumentsChain, CombineDocumentsChain) that automate this workflow [20].

Phase 1: The Map Step

During the map phase, the system treats each individual text chunk (or semantic cluster) as an isolated processing

unit [21]. The LLMChain maps a specific prompt over every single document chunk independently [20]. The foundation model is instructed via prompt engineering to act as an expert university professor, extracting core definitions, formulas, and code snippets while strictly adhering to the facts present in the text [1].

A standard prompt template utilized in the LangChain map step is defined programmatically as: [22]

Python

```
from langchain_core.prompts import PromptTemplate
prompt_template = """Write a concise summary of the following: "{text}" CONCISE SUMMARY:"""
PROMPT = PromptTemplate(template=prompt_template, input_variables=["text"])
```

The LLM generates a localized summary for each chunk, essentially translating highly verbose or messy student notes into clean, distilled insights, creating a set of optimized building blocks [21].

Phase 2: The Reduce Step and Hierarchical Merging

Once all individual chunks are summarized, the resulting texts are collected and passed into a Reduce Documents Chain [20]. This chain takes the individual summaries generated in the Map step and combines them into a single, cohesive text string [21].

However, for particularly dense academic modules with dozens of uploaded files, even the combined summaries might exceed the final LLM context window. In these scenarios, the reduce step must be executed recursively—a state-of-the-art process known as Hierarchical Merging [23].

In Hierarchical Merging, the level-0 summaries (from the initial map step) are grouped into sequentially logical pairs or batches, and the LLM is prompted to merge them into level-1 summaries [24]. This recursive consolidation continues through multiple layers until the entire corpus collapses into a single, definitive document [1]. Research tracking book-length document summarization indicates that hierarchical merging, while computationally heavier due to the increased number of API calls, matches or outperforms full-context processing in preserving structural integrity [25].

Multi-Agent and Hierarchical Multi-LLM Frameworks

To push beyond standard linear map-reduce limitations, SCHOLAR integrates architectures inspired by advanced Multi-Agent frameworks. Rather than using a single model across all domains, hierarchical multi-LLM summarization employs specialized AI agents to handle distinct components of the document. For instance, one agent handles formatting and extracting raw mathematical proofs, while another focuses purely on summarizing qualitative theory. These modular outputs are dynamically fused into a cohesive narrative through a test-time scalable multi-agent pipeline, which significantly improves both the factual accuracy and structural coherence of highly complex technical documents.

Enforcing Syllabus Guardrails

To prevent the LLM from hallucinating or losing thematic focus during extreme compression, SCHOLAR employs "Syllabus Pathing." Faculty coordinators can upload a master syllabus checklist for the specific course module [1]. This syllabus acts as a rigid schema regex constraint. The CombineDocumentsChain cross-references its generated outputs against the syllabus nodes, ensuring that all required topics are covered, extraneous tangents are discarded, and the final guide is formatted cleanly using Markdown tables and code blocks [1].

Retrieval-Augmented Generation (RAG) for Interactive Tutoring

While the generated "Perfect Note" resolves the issue of disorganized, fragmented study materials, technical students also require real-time, context-aware support to unpack complex engineering concepts. Standard parametric LLMs lack knowledge of specific, proprietary university course materials, recent lecture discussions, and professor-specific definitions [1]. Consequently, they frequently hallucinate when asked niche technical questions.

To solve this, SCHOLAR integrates an interactive conversational assistant directly within the native Android application interface, powered by an optimized Retrieval-Augmented Generation (RAG) architecture [1].

Theoretical Foundations of RAG

The RAG framework was introduced comprehensively to the NLP community by Lewis et al. in their 2020 NeurIPS paper, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" [26]. RAG models solve the hallucination problem by combining pre-trained parametric memory (the massive billions of weights inside the LLM) with non-parametric memory (a dynamically accessible, external vector corpus) [26].

By fine-tuning seq2seq models to condition their generation strictly on retrieved passages, RAG sets state-of-the-art benchmarks on open-domain question answering [26].

On-Device AI via Gemini Nano and AICore

To drastically reduce inference latency and address user privacy concerns, SCHOLAR leverages Android 14's AICore system service to access Gemini Nano. Gemini Nano allows the execution of powerful generative AI models directly on the mobile hardware, eliminating the necessity to transmit sensitive personal or academic queries to the cloud. Optimized in highly efficient formats such as the E2B model (optimized for maximum speed) and the E4B model (designed for complex reasoning tasks), the AICore infrastructure allows the system to synthesize answers directly on the edge. This saves battery life and guarantees that the interactive tutor functions rapidly, even in low-connectivity university environments.

Enterprise and Edge Vector Indexing

The retrieval pipeline is orchestrated through a hybrid network. Heavy backend jobs utilize Pinecone to navigate the collective university data, relying on advanced algorithms like Locality Sensitive Hashing (LSH) and Inverted File Index (IVF) [1,17].

Simultaneously, for isolated private study materials, the Kotlin mobile client utilizes ObjectBox.

This dual-indexing strategy guarantees that whether a student is searching across thousands of public course chunks or interacting strictly with their own downloaded, private notes, the corresponding approximate nearest neighbor (ANN) search operates efficiently without application lag.

Theoretical Background: Foundation Models and the Transformer Architecture

The entire deduplication, synthesis, and retrieval orchestration of SCHOLAR relies implicitly on the immense cognitive capabilities of underlying foundation LLMs [1]. The exponential leap in modern NLP capabilities was catalyzed by the 2017 introduction of the Transformer architecture by Vaswani et al. in their seminal paper "Attention Is All You Need" [2].

Prior to 2017, sequence-to-sequence (Seq2Seq) tasks like machine translation relied on Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks. These legacy architectures processed text sequentially (word-by-word), making them extremely slow to train and prone to catastrophic forgetting over long context windows [27].

The Transformer dispensed entirely with recurrence and convolutions, relying solely on an "Attention" mechanism [28]. The core mathematical innovation is "Scaled Dot-Product Attention," which allows the model to globally weigh the importance of different words in a sequence regardless of their physical distance from one another [27]. By calculating these attention scores in parallel across multiple independent "heads" (Multi-Head Attention), the model achieves profound semantic understanding of context, nuance, and syntax [27].

This deep, context-aware understanding enables the emergent reasoning capabilities observed in massive parameter models [12]. Within the SCHOLAR pipeline, these models do not merely execute rote pattern matching; they understand subtle paraphrases, complex academic analogies, and implied meanings within the engineering notes [12].

Empirical Evaluation, Benchmarking, and System Metrics

To quantitatively and objectively validate the quality of the AI-synthesized "Perfect Notes," the generated outputs were rigorously benchmarked against official university faculty textbooks and master slide resources using standardized, industry-grade NLP evaluation metrics—specifically BLEU and ROUGE [1].

BLEU Metric Alignment (Precision)

BLEU (Bilingual Evaluation Understudy) is an evaluation metric that measures precision against target reference texts [1]. It evaluates how many n-grams (sequences of consecutive words) generated in the AI summary also appear in the human-written reference document [29].

A critical component of the BLEU score is the Brevity Penalty (BP). If an AI simply outputs a single common word like "The", and that word appears in the reference text, a naive precision metric would award it a perfect score of 1.0. To mathematically punish "telegraphic" candidate strings that are too short to be meaningful, the brevity penalty is applied [30].

The mathematical expression for the Brevity Penalty is defined as:

$$BP = \begin{cases} 1 & \text{if } c > r \\ \exp(1 - \frac{r}{c}) & \text{if } c \leq r \end{cases}$$

Where c represents the length of the candidate AI translation/summary, and r represents the average length of the reference human translations [30].

The final BLEU score is the weighted geometric mean of all modified n-gram precisions, multiplied by the Brevity Penalty: [30]

$$\text{BLEU Score} = BP \times \exp \left(\sum_{i=1}^N w_i \ln(p_i) \right)$$

ROUGE Metric Evaluation (Recall)

While BLEU is inherently precision-oriented, ROUGE (Recall-Oriented Understudy for Gisting Evaluation) focuses heavily on recall [31]. It evaluates how much of the core information from the human-written reference is successfully captured and retained by the AI summary [1].

For ROUGE-N (evaluating unigrams, bigrams, etc.), the formula measures the overlap of n-grams against the total count in the reference: [32]

$$\text{ROUGE-N} = \frac{\sum_{n\text{-gram}} \min(\text{Count}_{\text{candidate}}, \text{Count}_{\text{reference}})}{\sum_{n\text{-gram}} \text{Count}_{\text{reference}}}$$

To capture more complex sentence structures, SCHOLAR specifically evaluates using ROUGE-L, which measures the Longest Common Subsequence (LCS) [1]. LCS identifies the longest sequence of words shared by both texts without requiring them to be strictly consecutive, allowing the metric to capture flexible word ordering, paraphrasing, and sequence structure [32].

The ROUGE-L F1-score balances both precision and recall using the harmonic mean, providing a highly robust, holistic assessment of the AI summary's structural integrity [33].

Quantitative System Benchmarks

During User Acceptance Testing (UAT), SCHOLAR achieved exceptional quantitative and qualitative results across sample B.Tech data science course evaluations [1].

Course Target Sample	Input Sources	ROUGE-L	BLEU	Human Accuracy Score (1-5)
Machine Learning Mod I	3 PDFs, 1 PPTX	0.84	0.79	4.7 / 5.0
Big Data Systems Mod III	4 Handwritten Notes	0.76	0.71	4.4 / 5.0
Deep Learning Overview	2 PDFs, 5 Images	0.81	0.75	4.6 / 5.0
1				

As demonstrated in the evaluation table, the pipeline effectively retains deep conceptual alignment with faculty textbooks. Double-blind scoring from academic staff confirmed that the semantic core, structural organization, and deduplication quality of the lecture were preserved, validating the platform's utility as a robust educational tool [1].

Operational Scalability and Hardware Constraints

To ensure operational feasibility during high-stress exam periods—when student traffic spikes exponentially—SCHOLAR was subjected to rigorous non-functional testing simulating massive concurrent loads [1].

Processing Latency Profiling

A major bottleneck in multi-document synthesis is the linear scaling of LLM processing times relative to document length and chunk volume. System testing tracked processing times against raw input sizes [1].

- 5 Pages: 10.1 seconds
- 10 Pages: 18.2 seconds
- 20 Pages: 31.0 seconds
- 30 Pages: 42.0 seconds [1]

The target operational baseline constraint (NFR-1) mandated that synthesizing a 50-page highly dense input cluster must finish within ≤ 45 seconds [1]. By employing asynchronous Python background worker queues and smart localized text block caching, the system remains within operational comfort zones, successfully preventing HTTP timeout requests from dropping the mobile client session [1].

RAG Concurrency and Latency

Conversational Time-To-First-Token (TTFT) for the RAG assistant was strictly mandated to stream within ≤ 1.5 seconds (NFR-2) [1]. Under automated JMeter spike load testing simulating 200 concurrent user sessions per engineering department (NFR-3), the hybrid vector retrieval network maintained a 96% accuracy grounding metric [1]. Even under maximum concurrent load, the vector spatial indexing and mobile network data streams exhibited no substantial degradation, ensuring high-fidelity, real-time tutoring support without application failures [1].

Future Trajectories: Agentic Frameworks and Expanding Architectures

While the current deployment of SCHOLAR operates as a highly optimized orchestrator, educational technology is rapidly advancing toward more autonomous, agentic reasoning models capable of highly complex problem-solving [1].

Tree of Thoughts (ToT) Integration

Current LLM inference is fundamentally confined to token-level, left-to-right decision-making processes [34]. For highly complex engineering tasks—such as debugging nested algorithmic code or balancing complex multi-variable physics equations—standard sequential prompting strategies frequently fall short, leading to cascading logic errors [34].

To surmount these challenges, researchers introduced the “Tree of Thoughts” (ToT) framework [34]. ToT provides a significantly superior reasoning structure by framing problems as a search over a combinatorial problem-space [34]. In the ToT paradigm, the LLM explores multiple different reasoning paths (represented as branches in a tree) over coherent units of text or mathematical steps (represented as nodes or “thoughts”) [34]. The model actively self-evaluates these partial solutions using generated heuristics, looking ahead or backtracking via breadth-first search (BFS) or depth-first search (DFS) to make optimal global choices [34]. Future iterations of SCHOLAR’s interactive RAG tutor are slated to implement ToT architectures [1]. This would allow the AI assistant to solve complex mathematical proofs alongside the student in a deliberate, highly reliable manner, dramatically reducing logical hallucinations [34].

Multimodal Parsing and Knowledge Graphs

Future architectural enhancements also include the integration of Knowledge Graph-enhanced Retrieval-Augmented Generation (KG-RAG) [35]. While pure semantic cosine similarity is incredibly fast, it relies strictly on vector proximity and occasionally misses crucial, explicit conceptual relationships that span across different courses or semesters [35]. By introducing directional semantic graph orchestration alongside vector embeddings, SCHOLAR could automatically map related topics across entire engineering departments [35]. This would allow the RAG tutor to autonomously pull in foundational physics concepts when a student is struggling with an advanced thermodynamics module, creating a truly holistic tutoring environment [1]. Furthermore, transitioning from standard OCR pipelines to native multimodal foundation models will allow the system to directly parse complex diagram sheets, flowcharts, and even lecture video streams without relying on intermediary, lossy text extraction [1]. This preserves the absolute spatial, chronological, and contextual integrity of visual data, elevating the synthesized notes from standard text documents to fully interactive, multimodal study guides [36].

Conclusion

The orchestration of fragmented, chaotic academic materials into a cohesive, centralized learning environment represents a profound leap forward in the domain of computer-supported collaborative learning. The architectural development and successful deployment of SCHOLAR demonstrates that coupling advanced Natural Language Processing pipelines with robust software engineering patterns can successfully, and demonstrably, mitigate the severe cognitive overload experienced by undergraduate engineering students.

By utilizing multi-format document ingestion pipelines, LayoutLM-inspired spatial awareness algorithms, and recursive character chunking, the platform reliably normalizes and standardizes highly unstructured data formats—including notoriously difficult handwritten notes. The subsequent application of Siamese BERT-Networks enables hyper-efficient semantic deduplication via mathematically rigorous cosine similarity matching, stripping away redundant peer data that traditionally clutters collaborative spaces. Ultimately, the integration of advanced Map-Reduce paradigms and Multi-Agent frameworks ensures that massive, book-length volumes of text are synthesized into authoritative, syllabus-aligned study guides without ever exceeding foundation model token constraints.

Concurrently, the deployment of a low-latency Retrieval-Augmented Generation (RAG) assistant—bolstered by on-device AI integration via Gemini Nano and ObjectBox local vector stores—directly addresses the limitations of passive, static study materials. By anchoring LLM inference heavily against localized, institution-specific vector embeddings, the system functionally eliminates hallucinatory responses, protects user privacy, and provides instantaneous, highly factual tutoring.

Backed by rigorous empirical validation using BLEU and ROUGE-L n-gram alignment metrics, the system proves that AI-driven orchestration can drastically reduce administrative preparation friction. By automating the tedious, manual workflows of exam revision prep, SCHOLAR ensures students remain focused entirely on active learning, deep comprehension, and academic mastery.

References

1. Project Report MAIN_MinorProject (2).pdf
2. Akcigit, U., Chikis, C. A., Dinlersoz, E., & Goldschlag, N. (2026). *Attention (and Money) is all you need: Why universities are struggling to keep AI talent* (No. w34964). National Bureau of Economic Research.
3. Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., & Zhou, M. (2020, August). Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1192-1200).
4. Corcoran, E. S. (2023). Note-taking, Center for the Advancement of Teaching Excellence, University of Illinois Chicago.
5. Thomas, L. (2023). 3 Collaborative Strategies to Build Up Students' Note-Taking Skills.
6. Zhang, Y. (2025). Student engagement in collaborative learning with AI agents in an LLM-empowered learning environment: A cluster analysis.
7. Sayeed, S. (2025). CollaClassroom: An AI-Augmented Collaborative Learning Platform with LLM Support in the Context of Bangladeshi University Students.
8. Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., & Zhou, M. (2020, August). Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1192-1200).
9. Xu, Y. (2020). BordiaS/layoutlm – GitHub.
10. Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., & Zhou, M. (2020, August). Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1192-1200).
11. Rezgui, I. (2025). The Ultimate Guide to RAG Chunking Strategies.
12. Kim, J., & Park, Y. (2026). A Survey of Text Deduplication: From Syntactic Matching to Semantic Understanding. *IEEE Access*.
13. Chen, Y., Yan, X., Meliou, A., & Lo, E. (2025). DiskJoin: Large-scale Vector Similarity Join with SSD. *Proceedings of the ACM on Management of Data*, 3(6), 1-27.
14. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019, June). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)* (pp. 4171-4186).
15. Nils, R. (2020). sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing* (pp. 3982-3992).
16. Reimers, N., & Gurevych, I. (2019, November). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)* (pp. 3982-3992).
17. Nearest Neighbor Indexes for Similarity Search
18. Cohen, R. S. (2023). Vector Similarity Explained
19. Kumar, G. (2024). Leveraging LangChain MapReduce: Summarize Large Texts and Translate into Any Language.
20. MapReduceDocumentsChain | langchain_classic - LangChain Reference Docs
21. Unlocking Legal Insights: Effortless Document Summarization with OpenAI's LLM and LangChain - R Systems
22. Reading LangChain's Summarization Code (2) - Map Reduce
23. Ou, L., & Lapata, M. (2025, July). Context-aware hierarchical merging for long document summarization. In *Findings of the Association for Computational Linguistics: ACL 2025* (pp. 5534-5561).
24. Chang, Y., Lo, K., Goyal, T., & Iyyer, M. (2024, May). Boookscore: A systematic exploration of book-length summarization in the era of llms. In *International Conference on Learning Representations* (Vol. 2024, pp. 56059-56093).
25. Chang, Y., Lo, K., Goyal, T., & Iyyer, M. (2024, May). Boookscore: A systematic exploration of book-length summarization in the era of llms. In *International Conference on Learning Representations* (Vol. 2024, pp. 56059-56093).
26. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33, 9459-9474.

27. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
29. Chiusano, F. (2022). Two minutes NLP — Learn the BLEU metric by examples.
30. BLEU - Wikipedia
31. ROUGE (metric) (2004)
32. BLEU and ROUGE score for NLP evaluation (17 Mar 2026)
33. Mamdouh, M. (2023). Mastering ROUGE Matrix: Your Guide to Large Language Model Evaluation for Summarization with Examples. *DEV Community*. Retrieved April, 30, 2025.
34. Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., & Narasimhan, K. (2023). Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36, 11809-11822.
35. Dong, C., Yuan, Y., Chen, K., Cheng, S., & Wen, C. (2025, March). How to build an adaptive AI tutor for any course using knowledge graph-enhanced retrieval-augmented generation (KG-RAG). In *2025 14th international conference on educational and information technology (ICEIT)* (pp. 152-157). IEEE.
36. Achkar, P., Gollub, T., & Potthast, M. (2025). Ask, retrieve, summarize: A modular pipeline for scientific literature summarization. *arXiv preprint arXiv:2505.16349*.
37. Sixu, A. N., Yu, Y. A. N. G., Yunsu, M. A., & Guandong, X. U. (2024, November). Developing an llm-empowered agent to enhance student collaborative learning through group discussion. In *International Conference on Computers in Education*.
38. Anderson, E., Lin, G. C., Farid, A., Fenech, M., Hanks, B., Klopfer, E., ... & Nguyen, A. (2025). Exploring GenAI technologies within collaborative learning.
39. Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., & Zhou, M. (2020, August). Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1192-1200).
40. Ciancone, M., Varangot-Reille, C., & Schaeffer, M. (2025). ChunkNorris: A High-Performance and Low-Energy Approach to PDF Parsing and Chunking. *arXiv preprint arXiv:2602.00010*.
41. Recursive Splitter | Project Lakechain
42. Shilov, I., Meeus, M., & de Montjoye, Y. A. (2025). The mosaic memory of large language models. *Nature Communications*, 17(1), 2142.
43. Abbas, A., Tirumala, K., Simig, D., Ganguli, S., & Morcos, A. S. (2023). Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*.
44. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019, June). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)* (pp. 4171-4186).
45. Nils, R. (2020). sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing* (pp. 3982-3992).
46. Nils, R. (2020). sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing* (pp. 3982-3992).
47. Tripathi, R. (2023). What is Similarity Search?
48. Bagaria, A. (2025). Clustering-Based Text Summarization Using LangChain's Map-Reduce Method.
49. How can LangChain be used to automate document summarization tasks?
50. Laforge, G. (2024). Summarization techniques, iterative refinement and map-reduce for document workflows
51. Sojasingarayar, A. (2024). Summarization with LangChain. Stuff — Map_reduce — Refine
52. Ou, L., & Lapata, M. (2025, July). Context-aware hierarchical merging for long document summarization. In *Findings of the Association for Computational Linguistics: ACL 2025* (pp. 5534-5561).
53. Master LLM Summarization Strategies and their Implementations
54. Ye, Z. (2025, February). Intelligent tutoring agent with retrieval-augmented generation: a case study of quality management system course. In *2025 5th International Conference on Consumer Electronics and Computer Engineering (ICCECE)* (pp. 189-194). IEEE.
55. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33, 9459-9474.
56. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33, 9459-9474.
57. Liu, L., & Cheng, Y. (2025, May 16). MinHash LSH in Milvus: The secret weapon for fighting duplicates in LLM training data. Milvus.
58. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

59. NLP - BLEU Score for Evaluating Neural Machine Translation
60. Chiusano, F. (2022). *Two Minutes NLP: Learn the ROUGE Metric by Examples*.
61. Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., & Narasimhan, K. (2023). Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36, 11809-11822.
62. Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., & Narasimhan, K. (2023). Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36, 11809-11822.
63. Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., & Narasimhan, K. (2023). Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36, 11809-11822.
64. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33, 9459-9474.