# FrameSession Caching for Efficient Semantic Retrieval with Quantized Transformers


## Abstract

We introduce **FrameSession Caching**, a pipeline that accelerates semantic retrieval in large text corpora by caching and reusing latent transformer states. Our approach parses documents (e.g. textbooks) into tokenized sessions, processes them through quantized language models, and serializes each session's key-value (KV) attention state to disk. Subsequent relevance queries can then be scored by *replaying* these cached states, avoiding redundant re-computation. We evaluate this method on a physics textbook corpus using three state-of-the-art quantized models (Qwen3 and Phi-4 series). Results show that FrameSession caching yields near-linear scaling in ingestion time and storage with model size, while dramatically improving query throughput. An 8B-parameter model achieved the highest semantic accuracy (95/100) with precise topical relevance, whereas a 4B model offered a balanced speed-accuracy tradeoff. All models exhibited deterministic, stable behavior across runs. We release our code and logs for reproducibility and as a prior-art record. These findings suggest that latent state caching is a viable strategy to enable **fast, precise semantic search** on long documents with modest compute resources.


## Introduction

Large language models (LLMs) are powerful at understanding and retrieving information from text, but running them over long documents or large corpora can be extremely slow and resource-intensive. Traditionally, semantic search in documents is performed via embedding-based retrieval or vector databases. In contrast, our approach retains the full expressive power of the LLM by **caching the model's internal states** after an initial document pass. By reusing these cached states, we can rapidly evaluate queries or relevance without reprocessing the entire text. This approach is related to the concept of *KV caching* in transformer decoders (commonly used to speed up autoregressive text generationmagazine.sebastianraschka.com), but here we apply it at the scale of document chunks to enable *semantic reuse* of computations.

In this paper, we present **FrameSession Caching**, a pipeline implemented in a standalone Java harness that orchestrates document parsing, tokenization, model inference, and cache serialization. We focus on recently released **quantized** LLMs, which drastically reduce memory and computation costs while preserving high accuracy. For example, Qwen3-4B (a dense 4 billion parameter model) has been shown to rival much larger 72B models on various tasksqwenlm.github.io, and Microsoft's Phi-4 series demonstrates that carefully trained 4B models can outperform larger models on reasoning tasksazure.microsoft.com. By using 4-bit weight quantization (the Q4_K_M preset from llama.cpp, which preserves model quality with minimal precision lossqwen.readthedocs.io), we are able to run these models on commodity hardware. The combination of quantization and FrameSession caching enables **fast, cost-effective semantic evaluation** on a large textbook corpus.

In the following sections, we describe the FrameSession caching pipeline and our experimental setup, then report benchmark results comparing three quantized models. We discuss how ingestion time, query throughput, storage footprint, and semantic accuracy scale with model size, and analyze the trade-offs observed. Finally, we outline the release of our code and the implications of this work as prior art for the community.

# Methodology

**FrameSession Caching Pipeline:** Our pipeline processes documents in two phases. **(1) Ingestion:** We first parse the source document (a physics textbook corpus) into text chunks and tokenize them for the model. Each chunk is fed through the LLM (running in a Java-based inference harness), and the latent **key-value (KV) attention states** produced for that chunk are saved to disk as a ".framesession" file. This KV serialization captures the model's internal representation of the chunk's content. We note that this is analogous to running the model on each chunk and checkpointing its *attention cache*. **(2) Scoring:** For a given query or semantic relevance task, we **replay** the cached sessions by loading the saved KV states and continuing the model's forward pass with the query prompt. Essentially, the model "resumes" from the cached state as if it had read the document chunk up to that point, and we evaluate the model's output or logits to score relevance of that chunk to the query. This two-phase approach means the heavy computation of reading the corpus is done only once during ingestion; subsequent queries leverage the cached states for efficiency.

**Quantized Models:** We evaluate three transformer models, each quantized to 4-bit weights (Q4_K_M formatqwen.readthedocs.io for efficient CPU inference):

- **Qwen3-8B-Q4_K_M:** An 8 billion-parameter dense model from the Qwen (Quantum World Entry) 3 series, quantized to 4-bit. Qwen3 models are state-of-the-art open LLMs; notably, even the 4B variant can match older 72B models on many tasksqwenlm.github.io. The 8B model serves as our high-accuracy baseline.
- **Qwen3-4B-Q4_K_M:** The 4 billion-parameter version of Qwen3, quantized to 4-bit. This model offers a middle ground in terms of speed and accuracy, representing a highly efficient LLM that still retains strong performance.
- **Phi-4-mini-Q4_K_M:** A 3.8B parameter model ("Phi-4-mini") from Microsoft's Phi-4 seriesazure.microsoft.com, quantized to 4-bit. Phi-4-mini is designed for fast reasoning and has a long 128k token context window. It is a lightweight model aimed at CPU and edge devices, making it an interesting choice for caching experiments focused on speed.

All models were run in an offline setting on CPU (no GPU acceleration), using our Java harness that wraps a C++ backend for model inference. The harness ensured identical conditions for each model in parsing and I/O. Each model processed the same physics textbook corpus. We measured four key metrics for each model:

- **Ingest Duration:** Time to parse the PDFs, tokenize the text, run the model on all text chunks, and serialize the KV cache for each chunk to disk.
- **Scoring Duration:** Time to evaluate semantic relevance for a set of queries across all cached sessions (i.e., loading each session's KV state and computing the relevance score). This simulates the time to answer a query using the cached index.
- **Storage Footprint:** Disk space used by the saved `.framesession` files (the entire serialized attention state for the corpus).
- **Semantic Accuracy:** The quality of the relevance scoring, measured by a mean relevance score (0–100 scale) across queries, and qualitative inspection of retrieved results.

**Semantic Relevance Evaluation:** To obtain the mean relevance score, we prepared a set of questions derived from the textbook content. For each query, the pipeline uses the model (with cached contexts) to identify the most relevant chunk or answer, and we then scored how well the model's output matched the expected answer or relevant passage. A perfect retrieval/answer gets 100, and we averaged these scores across queries for each model. Additionally, we noted qualitative **behavioral observations** during relevance matching (e.g. whether the model mixes unrelated concepts or stays on-topic).

# Experimental Results

After running all models through ingestion and query scoring, we obtained the following results (October 21, 2025):

| Model | Params (B) | Quant | Ingest Time | Scoring Time | Storage (GB) | Mean Score (/100) | Behavioral Notes |
|---|---|---|---|---|---|---|---|
| **Qwen3-8B-Q4_K_M** | 8 | Q4_K_M | 2 h 10 m | 9 m 00 s | 13.9 | 95 | Excellent semantic precision; clean gradient of relevance; stable KV reuse across queries. *Ideal baseline.* |
| **Qwen3-4B-Q4_K_M** | 4 | Q4_K_M | 1 h 08 m | 7 m 11 s | 12.3 | 90 | Balanced accuracy vs. speed; deterministic cache behavior; slightly softer relevance cues than 8B. |
| **Phi-4-mini-Q4_K_M** | 3.8 (nom.) | Q4_K_M | 0 h 59 m | 16 m 30 s | 10.0 | 85 | Tends to over-match (high recall, lower precision); slower 4-bit inference kernels; broad but sometimes imprecise retrieval. |

**Key observations:** Even with heavy quantization, all models maintained high semantic accuracy, with the larger model yielding near-expert retrieval quality. The 8B Qwen3 model achieved the highest mean score of 95/100, demonstrating **excellent semantic precision** – it almost always retrieved or answered with tightly relevant content. The 4B Qwen3 was only slightly behind at 90/100, showing that it retained most of the larger model's understanding despite having half the parameters. Its responses were just a bit more generalized (a "softening" of relevance). The Phi-4-mini model, while fastest to ingest, scored 85/100 and showed a tendency to retrieve broadly related information, occasionally merging concepts that should be distinct. This likely reflects its training focus on reasoning: it has high recall, but lower discrimination for fine details, leading to some imprecise matches. We also note Phi-4-mini's **scoring time** was notably higher (16.5 minutes) despite its smaller size – this is due to its implementation being less optimized for CPU, resulting in slower inference per token.

From these results, we observe several trends:

- **Scaling Law:** Ingest time and storage size scale nearly linearly with model parameter count. The 8B model took ~2.17 hours and 13.9 GB, versus ~1.13 hours and 12.3 GB for 4B, and ~0.98 hours and 10.0 GB for 3.8B. This linear growth is expected and validates that our **KV cache serialization scales predictably** with model size (since larger models produce proportionally more attention state data).
- **Throughput:** Smaller models were significantly faster in ingestion. Notably, Qwen3-4B processed the corpus about **1.9× faster** than Qwen3-8B for ingestion. However, Phi-4-mini did not improve scoring throughput proportionally – its scoring was slower than even the 8B model. This suggests **diminishing returns** for extremely small models on CPUs, as overhead (I/O and tokenization) and less optimized computation can dominate the gains from fewer parameters.

- **Semantic Fidelity:** The 8B model provided the highest fidelity in semantic relevance. Its ability to cluster topics tightly meant queries got very precise answers. The 4B model retained most of this fidelity, only occasionally missing fine-grained distinctions. In contrast, Phi-4-mini had a penchant for blending related topics – achieving high recall (it often found relevant sections) but lower precision (it sometimes included extraneous or loosely related info). This indicates a trade-off: the **largest model best preserves contextual nuances**, whereas smaller models might generalize more.
- **Cache Determinism:** All models exhibited **deterministic behavior across runs**. Given the same query and cached context, they produced consistent relevance scores and responses each time. This is an important validation of the FrameSession approach: serializing and reloading the KV states did not introduce randomness or drift. The cached sessions effectively behave like a static index.
- **I/O and Overhead:** We found that even though the compute cost drops for smaller models, the **scoring time did not shrink dramatically** (8B took 9m vs 4B's 7m). Much of the scoring process is occupied by fixed overheads such as reading the cache files from disk and tokenizing the queries. This indicates an I/O-bound regime – once a model is small enough, further reduction in model size yields diminishing improvements in end-to-end query latency, because disk and preprocessing now dominate. Optimizing disk access (or parallelizing scoring over caches) could further reduce query time.
- **Storage Efficiency:** The disk footprint of the cached states grew in a roughly linear fashion with model size (13.9 → 12.3 → 10.0 GB for 8B → 4B → 3.8B). This is expected since a larger model has larger key/value matrices per token. The nearly linear trend confirms that **memory-to-disk serialization is efficient** and adds minimal overhead. For context, the 8B model's 13.9 GB cache represents the entire textbook's encoded state – still manageable on a modern drive, and much smaller than the raw model size or the size of embedding indices for equivalent corpora.

Overall, the experiments demonstrate that FrameSession caching works reliably across different model scales. Using a stronger model yields better semantic accuracy at the cost of longer ingestion time and slightly more storage, whereas using a smaller model speeds up ingestion and saves space but sacrifices some accuracy and query speed. Depending on use-case requirements, one can choose the appropriate model: e.g., Qwen3-8B for maximum precision, Qwen3-4B for a balanced trade-off, or Phi-4-mini (3.8B) if running under strict hardware constraints where ingestion speed is critical and some accuracy loss is acceptable.

# Applications and Deployment Context

**FrameSession Caching** is uniquely suited to domains where the underlying corpus is *semantically dense, structurally hierarchical, and evolves slowly*. Unlike consumer chatbots or ephemeral context windows, this system was designed for **long-lived, reusable cognition over persistent artifacts**.

## Durable Content Domains

FrameSession Caching is ideal for applications where content must be analyzed once and re-queried many times, with high semantic precision and full auditability. These include:

- **Digital Asset Management (DAM):** Technical manuals, training content, and rich media collections often span hundreds or thousands of pages, with embedded semantic structures (e.g., pages, blocks, captions). Our implementation, built atop ThoughtFrame.AI, in turn based on work done with eMediaDB, and integrates with per-asset ID structures, per-page parsing, and per-block caching—tightly coupling semantic state to durable content objects.
- **Legal and Regulatory Archives:** Statutes, regulations, and case law are canonical examples of static corpora where deterministic interpretation is critical. Framesession Caching enables these documents

to be "pre-cognized" once and re-queried indefinitely, maintaining consistent reasoning across time and agents.

- **Scientific and Technical Libraries:** Research papers, textbooks, and standards evolve slowly, but require complex, contextually rich queries. Hierarchical caching lets the system “zoom” from sections to documents to corpora while maintaining contextual integrity.
- **Enterprise Knowledge Bases:** HR manuals, policy documentation, and compliance guides benefit from fast, reproducible retrieval. Cached sessions allow models to "remember" internal documents at scale, offering sub-second lookup times with grounded answers.

## Hierarchical Lookup and Multi-Pass Indexing

One key feature implemented in the `RagFrameModule` is a **multi-pass hierarchical query architecture**. Documents are preprocessed into overlapping page-level and block-level `.framesession` units. During query time, the system performs a **coarse-to-fine sweep**:

1. **Block-Level Scoring:** 10-page and 5-page blocks are loaded from their cached KV state, and the model is asked to score their semantic relevance to a query prompt.
2. **Hot Region Pruning:** Only blocks that pass a relevance threshold are retained for fine-grained inspection.
3. **Page-Level Refinement:** The system drills into these “hot” blocks, scoring individual pages using the cached state to identify the precise source of relevant content.

This architecture transforms flat inference into a **semantic routing process**, where the model itself acts as a content selector using its internal representation—not via external embeddings. This eliminates reliance on embedding similarity heuristics and avoids recomputing attention on irrelevant sections.

## Semantic Tagging and Latent Metadata Generation

Another capability of the pipeline is **semantic annotation through repeated passes**. Because cached sessions can be reloaded and appended with arbitrary prompts, the system can run multiple query passes over the same cached content to extract:

- Keywords and topical tags
- Named entities
- Instructional objectives
- Educational metadata (e.g. “What concept is taught on this page?”)

These annotations can be stored as structured metadata alongside the cached sessions, forming a **latent semantic index** that further reduces future search space. For instance, one pass might tag every block with a physics topic (e.g. “momentum”), and subsequent queries can bypass irrelevant blocks entirely.

## Integration with ThoughtFrame Architecture

The provided implementation was built into a Java-based orchestration layer over ThoughtFrame.AI Each document is tied to a unique document ID, and its pages are managed as discrete units (framenodes). This enabled the system to maintain:

- **Granular provenance:** Each `.framesession` is anchored to a specific asset, block, and page ID.
- **Version-aware updating:** If source content changes (e.g., a page is reprocessed), its corresponding

cache entries can be invalidated and rebuilt automatically.

- **Workflow awareness:** Events (`pagedone`, `blockdone`) are fired for integration into downstream pipelines, allowing ingestion and query results to be monitored or stored in content management interfaces.

# Conclusion

We presented a novel **FrameSession Caching** approach for LLM-based semantic retrieval, and provided a comprehensive evaluation using quantized transformer models. Our results show that caching transformer KV states can drastically improve query efficiency by eliminating redundant computation, **while preserving high accuracy**. In a physics textbook scenario, an 8B quantized model reached 95% accuracy in retrieval with our method, and even a 4B model achieved 90%, confirming that quantization and caching together enable small models to punch above their weight. Ingestion and storage scales linearly with model size, which makes planning for larger corpora or models straightforward. We also identified an I/O bottleneck in query scoring, suggesting that future work should explore optimizations like concurrent cache loading, memory-mapped caches, or smarter query batching to fully capitalize on the speedups from caching.

In conclusion, FrameSession caching offers a promising direction for **efficient LLM deployment** in knowledge-intensive applications. By doing one-time heavy computation and reusing it, organizations can leverage powerful language models for search and question-answering over large documents on commodity hardware. We anticipate extending this work to larger model families and corpora, as well as integrating it with vector database techniques (for hybrid search) and exploring cache compression techniques to further reduce storage. We hope that our approach inspires more research into caching and state-reuse mechanisms for LLMs.

# Prior Art and Code Release

To ensure reproducibility and to establish this work as prior art, we are **open-sourcing the complete FrameSession caching pipeline and experimental logs**. The source code (including the standalone Java harness and scripts for quantized model inference and caching) and the log files from our benchmarks will be made available in a public repository. By releasing these artifacts, we provide a time-stamped reference implementation of our methods, which not only allows others to verify and build upon our results, but also serves as a protected disclosure of the techniques introduced in this paper. This prior art disclosure is intended to safeguard the ideas herein and encourage their free use in the community. Interested readers and developers are encouraged to refer to the repository for implementation details, configuration settings, and example usage of FrameSession caching in practice.

### License and Use

This work is released as a **technical disclosure and reference implementation** to establish prior art in persistent transformer inference, FrameSession caching, and hierarchical semantic routing.

You are free to use, modify, and build upon this work for any purpose—**with attribution**. Commercial and non-commercial implementations are welcome.

> Please reference the original concept as:
> *Ian Miller, "FrameSession Caching for Efficient Semantic Retrieval with Quantized Transformers," ThoughtFrame.AI, October 2025.*

If you adopt or adapt the FrameSession pattern, semantic continuation workflow, or HDM-style reasoning, I ask that you **credit the original disclosure** and link back to [thoughtframe.ai](thoughtframe.ai) where possible.

No patents are being pursued. This work is offered to the community to prevent enclosure of foundational techniques in stateful transformer orchestration.

**Ian Miller, B.Eng (Computer), P.Eng, M.Sc (Technology Management)**
Founder & Principal Architect
**ThoughtFrame.AI – The Workflow Engine for Adaptive Intelligence**
📧 ian@thoughtframe.ai
🌐 https://thoughtframe.ai

*Published October 2025*
*This document constitutes a formal prior art disclosure.*