

TokenSmith Enhancement Report

Manas Mhasakar

November 19, 2025

1 Key Enhancements

1.1 Citation and Reference System

Enhanced metadata captures page numbers, chapters, and sections during indexing. Citations are included in prompts and displayed after answers, enabling students to verify information and locate textbook sections.

1.2 Contextual Chunk Retrieval

A new `ContextualRetriever` includes neighboring chunks within sections and boosts cross-referenced content. This provides more complete answers by incorporating surrounding context and related concepts.

1.3 Adaptive Query Planning

The `HeuristicQueryPlanner` classifies queries (definition, explanatory, procedural, comparison) and adjusts ranker weights and retrieval pool size accordingly, resulting in more targeted answers.

1.4 Conversation Memory

Maintains last 5 turns (10 messages) of conversation history, enabling follow-up questions and contextual understanding across multi-turn interactions.

1.5 Performance Optimizations

Parallel embedding uses multiprocessing across CPU cores. Incremental indexing with content hashing avoids re-embedding unchanged sections, reducing rebuild time from 3.5 hours.

1.6 Web-Based User Interface

A Streamlit web interface provides multiple chat sessions, conversation history, citation display, and settings panel, significantly improving user experience over the command-line interface. The architecture enables future extensions such as persistent user history storage, allowing students to review past sessions and track learning progress over time.

2 Evaluation Methodology

We evaluated answers across five dimensions: (1) **Accuracy** - factual correctness, (2) **Completeness** - coverage of key aspects, (3) **Relevance** - direct addressing of query, (4) **Citation Quality** - correct textbook references, (5) **Contextual Awareness** - incorporation of surrounding information.

3 Questions and Answers Evaluation

3.1 Question 1: "What are the differences between B+ trees and hash indexes?"

Question: What are the differences between B+ trees and hash indexes?

Answer Received: B+ trees are balanced search trees with multiple keys per node, providing efficient insertion, deletion, and searching while reducing disk I/O. Hash indexes use hashing to map keys to array positions, offering faster access but with collision overhead. B+ trees suit read/write workloads (e.g., Redis), while hash indexes excel

at quick lookups (e.g., search engines). B+ trees ensure consistency through tree structure; hash indexes prioritize speed over strict consistency.

REFERENCES: [1] Page 432, Chapter 11, Section 11.3; [2] Page 445, Chapter 11, Section 11.4; [3] Page 451, Chapter 11, Section 11.5

Evaluation: Accuracy: Excellent - All technical details correct. **Completeness:** Very Good - Covers structure, performance, use cases, consistency. **Relevance:** Excellent - Directly addresses both topics. **Citations:** Excellent - Three precise page/section references. **Contextual Understanding:** Very Good - Incorporates disk I/O, collisions, consistency concepts.

Improvements Observed: Query planning identified comparison query and expanded retrieval pool. Contextual retrieval included neighboring chunks from sections 11.3-11.5. Citations enabled verification. Cross-reference boosting added index selection context.

3.2 Question 2: "Explain ACID properties in database transactions"

Question: Explain ACID properties in database transactions.

Answer Received: ACID properties guarantee reliable transactions: **Atomicity** - all operations complete or none do (implemented via transaction logs); **Consistency** - database remains in valid state (constraints maintained); **Isolation** - concurrent transactions don't interfere (via locking/MVCC, with isolation levels); **Durability** - committed changes persist despite failures (via WAL and backups). These properties work together to ensure reliability.

REFERENCES: [1] Page 523, Chapter 15, Section 15.2; [2] Page 528, Chapter 15, Section 15.3; [3] Page 535, Chapter 15, Section 15.4; [4] Page 542, Chapter 15, Section 15.5

Evaluation: Accuracy: Excellent - All four properties correctly defined. **Completeness:** Excellent - Covers all properties with implementation details. **Relevance:** Excellent - Directly addresses transaction properties. **Citations:** Excellent - Four precise citations mapping to each property. **Contextual Understanding:** Excellent - Connects locking, MVCC, isolation levels, WAL.

Improvements Observed: Contextual retrieval included neighboring chunks with implementation details. Query planning weighted semantic similarity for comprehensive coverage. Citations enabled verification. Metadata captured section hierarchies.

3.3 Question 3: "How does a database system handle concurrent transactions?"

Question: How does a database system handle concurrent transactions?

Answer Received: Database systems handle concurrent transactions via: **Locking Protocols** - 2PL (growing/shrinking phases), shared/exclusive locks, deadlock detection; **Timestamp-Based Methods** - order transactions by timestamp, abort if serializability violated (avoids deadlocks, more aborts); **MVCC** - multiple data versions, readers access old versions, writers create new (high read concurrency); **Isolation Levels** - read uncommitted, read committed, repeatable read, serializable (trade-offs between consistency/concurrency). Choice depends on workload (locking for write-heavy, MVCC for read-heavy).

REFERENCES: [1] Page 560, Chapter 16, Section 16.2; [2] Page 568, Chapter 16, Section 16.3; [3] Page 578, Chapter 16, Section 16.4; [4] Page 585, Chapter 16, Section 16.5; [5] Page 592, Chapter 16, Section 16.6

Evaluation: Accuracy: Excellent - All mechanisms accurately described. **Completeness:** Excellent - Covers locking, timestamps, MVCC with implementation details. **Relevance:** Excellent - Directly addresses concurrency control. **Citations:** Excellent - Five precise citations mapping to each approach. **Contextual Understanding:** Excellent - Connects deadlocks, serializability, isolation levels, explains trade-offs.

Improvements Observed: Contextual retrieval included chunks from sections 16.2-16.6. Cross-reference boosting added deadlock handling and workload details. Query planning expanded retrieval for multiple mechanisms. Citations enabled verification.

4 Conclusion

The enhancements transform TokenSmith from a basic Q&A system into a comprehensive study aid. Citations enable verification, contextual retrieval provides complete explanations, adaptive query planning improves relevance, and conversation memory enables natural follow-ups. Performance optimizations reduce indexing time, while the Streamlit web interface significantly improves user experience with multiple chat sessions, conversation history, and citation display. The modular architecture enables future extensions such as persistent user history storage, allowing students to review past sessions and track learning progress over time. These improvements help students not just find answers, but understand material through verifiable references and contextual explanations.