**ASIST PERFORMANCE OPTIMIZATION STRATEGY**

**PROBLEM STATEMENT:**

**Existing App performance is extremely slow , user waiting more than 2 mins to generate answer and there is no streaming in place.**

**GOALS &STEPS:**

**Expected Improvement: Fixes should cut response time by ~50% (to ~15-20s) by eliminating the second Ollama generation pass.**

1. **Establish Baseline: Run test with your current code**
2. **Apply Fixes: Make the three changes identified BELOW**
3. **Re-test: Run again to measure improvement**
4. **Compare: The script saves JSON results for comparison**

**Expected Improvement: Fixes should cut response time by ~50% (to ~15-20s) by eliminating the second Ollama generation pass.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Results After All Fixes(without Cached layer added)** | | | |
|  |  |  |  |
| **Component** | **Before** | **After** | **Improvement** |
| Intent | 33s | 8-10s | 70% faster |
| Entity | 59s | 15-20s | 66% faster |
| Answer | 71s | 20-25s | 65% faster |
| **Total** | **163s** | **45-55s** | **68% faster** |

**Before fixing Which agency is responsible for Navy FMS cases?**

**[TIMING] \_analyze\_intent\_step: 33.11s**

**[TIMING] extract\_and\_retrieve: 58.92s**

**[TIMING] generate\_answer: 71.14s**

**[TIMING] process\_query: 163.17s**

**Issue: Model: llama3.2:latest at 2.0 GB**

**This is llama3.2:3b (3 billion parameters). It's a mid-size model that's causing your slowness. On CPU-only Windows, this explains your 163-second response times.**

**Why It's So Slow**

* **3b model on CPU: Each generation takes 30-70 seconds**
* **3 separate Ollama calls per query: Intent (33s) + Entity (59s) + Answer (71s)**
* **No GPU acceleration: Windows CPU-only inference is ~10x slower than GPU**

**Immediate Fix (Will Cut Time by 60-70%)**

**Step 1: Switch to the 1b model-** **A screen shot of a computer

AI-generated content may be incorrect.**

Step 2: .env file changes for ollama model

A screenshot of a computer

AI-generated content may be incorrect.

Step 3: Code Optimizations for app.py

**Fix 2: Reduce Token Generation (Expected: -30% time)**

Changed from A screenshot of a computer program

AI-generated content may be incorrect.

TO A screenshot of a computer program

AI-generated content may be incorrect.

B: **Second pass disabled** in \_generate\_with\_validation():

* <= 2 → > 10

FROM:

if validation\_results["needs\_improvement"] and len(validation\_results["issues"]) <= 2:

TO:

**Answer Length Impact on Performance**

Your current answer is probably **400-600 tokens**. With num\_predict=512:

* Llama generates ~8-12 tokens/second on CPU
* 500 tokens ÷ 10 tokens/sec = **50 seconds just for generation**

If you make answers more concise (200-250 tokens):

* Answer generation: 50s → **20-25s** (saves 25-30s)
* New total: 130s → **100-105s**

**How to Make Answers Concise**

**Option 1: Reduce max tokens further** In call\_ollama\_enhanced():

python

"num\_predict": 200 *# Change from 512 to 200*

A screenshot of a computer program

AI-generated content may be incorrect.

AFTER change 200

A screenshot of a computer program

AI-generated content may be incorrect.

**Adjust length guidelines** In EnhancedAnswerAgent.\_\_init\_\_(), reduce all targets:

**BEFORE**

A computer screen shot of a program

AI-generated content may be incorrect.

**AFTER**

**A screenshot of a computer code

AI-generated content may be incorrect.**

**CHANGE BASE INSTRUCTIONS**

**base\_instructions = {**

**"definition": """You are a SAMM (Security Assistance Management Manual) expert specializing in precise definitions.**

**TASK: Provide authoritative definitions with exact SAMM section citations.**

**CRITICAL REQUIREMENTS:**

**- Use exact SAMM terminology and definitions from the context**

**- Always cite specific SAMM sections (e.g., "SAMM C1.3.2.2")**

**- Expand acronyms on first use (e.g., "Defense Security Cooperation Agency (DSCA)")**

**- Distinguish between Security Cooperation and Security Assistance**

**- Remember: SA is a SUBSET of SC, not the same thing**

**- Provide clear, complete definitions that could stand alone**

**RESPONSE STRUCTURE:**

**1. Clear definition statement**

**2. SAMM section citation**

**3. Additional context about authority/oversight if relevant""",**

**"distinction": """You are a SAMM expert specializing in explaining key distinctions and differences.**

**TASK: Clearly explain differences between SAMM concepts with precise legal and operational distinctions.**

**CRITICAL REQUIREMENTS:**

**- Highlight key differences clearly and systematically**

**- Explain legal authority differences (Title 10 vs Title 22)**

**- Use specific examples when possible**

**- Always emphasize that Security Assistance is a subset of Security Cooperation**

**- Cite relevant SAMM sections for each concept being compared**

**- Address common misconceptions**

**RESPONSE STRUCTURE:**

**1. State the key distinction clearly**

**2. Explain each concept separately with citations**

**3. Highlight the differences with examples**

**4. Summarize the relationship""",**

**"authority": """You are a SAMM expert specializing in authority and oversight structures.**

**TASK: Explain who has authority, oversight, and responsibility for specific programs.**

**CRITICAL REQUIREMENTS:**

**- Clearly state which organization/person has authority**

**- Explain the scope of authority and oversight**

**- Cite legal authorities (FAA, AECA, NDAA, Executive Orders)**

**- Distinguish between "supervision," "direction," and "oversight"**

**- Reference specific SAMM sections**

**- Explain delegation chains where applicable**

**RESPONSE STRUCTURE:**

**1. State who has the authority**

**2. Explain the scope and basis of authority**

**3. Cite legal foundations**

**4. Describe any delegation or coordination requirements""",**

**"organization": """You are a SAMM expert specializing in organizational roles and responsibilities.**

**TASK: Describe organizations, their roles, and specific responsibilities.**

**CRITICAL REQUIREMENTS:**

**- Provide full organization names and acronyms**

**- List specific roles and responsibilities clearly**

**- Explain relationships between organizations**

**- Cite relevant SAMM sections**

**- Include key personnel authorities where applicable**

**- Describe organizational structure and reporting relationships**

**RESPONSE STRUCTURE:**

**1. Full name and acronym**

**2. Primary role and mission**

**3. Specific responsibilities**

**4. Reporting relationships and coordination""",**

**"factual": """You are a SAMM expert providing specific factual information.**

**TASK: Provide accurate, specific facts from SAMM Chapter 1.**

**CRITICAL REQUIREMENTS:**

**- Provide precise, accurate information**

**- Include dates, numbers, and specific details**

**- Cite SAMM sections for verification**

**- Use exact terminology from SAMM**

**- Expand acronyms appropriately**

**RESPONSE STRUCTURE:**

**1. Direct answer to the factual question**

**2. Supporting context**

**3. Source citation""",**

**"relationship": """You are a SAMM expert explaining relationships between entities and concepts.**

**TASK: Describe how SAMM entities, programs, and authorities relate to each other.**

**CRITICAL REQUIREMENTS:**

**- Clearly explain the nature of relationships**

**- Use specific examples to illustrate connections**

**- Cite relevant authorities and SAMM sections**

**- Explain the significance of relationships**

**- Address coordination and oversight aspects**

**RESPONSE STRUCTURE:**

**1. Describe the relationship clearly**

**2. Explain why the relationship exists**

**3. Provide examples of how it works in practice**

**4. Cite supporting authorities""",**

**"general": """You are a SAMM (Security Assistance Management Manual) Chapter 1 expert.**

**TASK: Provide comprehensive, accurate information about Security Cooperation and Security Assistance.**

**CRITICAL REQUIREMENTS:**

**- Use exact SAMM terminology from the provided context**

**- Always cite SAMM sections when available**

**- Expand acronyms on first use**

**- Maintain distinction between SC and SA (SA is subset of SC)**

**- Provide authoritative, accurate information**

**- Structure responses logically and completely"""**

**}**

**ADDING UP CACHED LAYER-STEP BY STEP APPROACH**

**Step 1:Add Cache Variables in IntegratedEntityAgent**

**def \_\_init\_\_(self, knowledge\_graph=None, db\_manager=None):**

**print("[IntegratedEntityAgent] Initializing with database connections...")**

**self.knowledge\_graph = knowledge\_graph**

**self.db\_manager = db\_manager or db\_manager**

**# Learning and feedback systems**

**self.hil\_feedback\_data = []**

**self.custom\_entities = {}**

**self.trigger\_updates = []**

**self.dynamic\_knowledge = {**

**"entities": {},**

**"relationships": []**

**}**

**# ADD THESE LINES HERE (after line 558):**

**# Cache system**

**self.entity\_cache = {}**

**self.cache\_max\_size = 100**

**self.cache\_hits = 0**

**self.cache\_misses = 0**

**# Enhanced entity patterns...**

**# (rest of existing code continues)**

**Step 2: Add Cache Helper Methods**

**(Inside IntegratedEntityAgent class , Before extract\_and\_retrieve method)**

**# ADD THESE METHODS HERE (before extract\_and\_retrieve at line 803):**

**def \_create\_cache\_key(self, query: str, intent\_info: Dict) -> str:**

**"""Create normalized cache key"""**

**normalized\_query = query.lower().strip()**

**intent = intent\_info.get('intent', 'unknown')**

**return f"{normalized\_query}|{intent}"**

**def \_add\_to\_cache(self, cache\_key: str, result: Dict):**

**"""Add to cache with size limit"""**

**if len(self.entity\_cache) >= self.cache\_max\_size:**

**first\_key = next(iter(self.entity\_cache))**

**del self.entity\_cache[first\_key]**

**print(f"[EntityCache] Removed oldest entry")**

**self.entity\_cache[cache\_key] = result.copy()**

**def clear\_cache(self):**

**"""Clear cache"""**

**self.entity\_cache.clear()**

**self.cache\_hits = 0**

**self.cache\_misses = 0**

**print("[EntityCache] Cleared")**

**def get\_cache\_stats(self) -> Dict:**

**"""Get cache statistics"""**

**total = self.cache\_hits + self.cache\_misses**

**hit\_rate = (self.cache\_hits / total \* 100) if total > 0 else 0**

**return {**

**"cache\_size": len(self.entity\_cache),**

**"max\_size": self.cache\_max\_size,**

**"hits": self.cache\_hits,**

**"misses": self.cache\_misses,**

**"hit\_rate": round(hit\_rate, 2)**

**}**

**Step 3: Modify extract\_and\_retrieve Method**

**Replace existing extract and retrieve method**

**@time\_function**

**def extract\_and\_retrieve(self, query: str, intent\_info: Dict) -> Dict[str, Any]:**

**"""Main method with caching"""**

**# CHECK CACHE FIRST**

**cache\_key = self.\_create\_cache\_key(query, intent\_info)**

**if cache\_key in self.entity\_cache:**

**self.cache\_hits += 1**

**print(f"[EntityCache HIT] hits:{self.cache\_hits}, misses:{self.cache\_misses}")**

**cached\_result = self.entity\_cache[cache\_key].copy()**

**cached\_result['from\_cache'] = True**

**return cached\_result**

**# CACHE MISS - Do full extraction**

**self.cache\_misses += 1**

**print(f"[EntityCache MISS] hits:{self.cache\_hits}, misses:{self.cache\_misses}")**

**print(f"[IntegratedEntityAgent] Processing query: '{query}' with intent: {intent\_info.get('intent', 'unknown')}")**

**# ORIGINAL CODE CONTINUES HERE (lines 808-888 ki existing logic)**

**try:**

**# Phase 1: Enhanced entity extraction**

**entities = self.\_extract\_entities\_enhanced(query, intent\_info)**

**print(f"[IntegratedEntityAgent] Extracted entities: {entities}")**

**# Phase 2: Query all data sources**

**all\_results = {**

**"query": query,**

**"entities": entities,**

**"intent\_info": intent\_info,**

**"timestamp": datetime.now().isoformat(),**

**"data\_sources": {},**

**"context": [],**

**"text\_sections": [],**

**"relationships": [],**

**"confidence\_scores": {},**

**"overall\_confidence": 0.0,**

**"extraction\_method": "integrated\_database\_enhanced",**

**"extraction\_phases": ["pattern\_matching", "nlp\_extraction", "database\_queries"],**

**"phase\_count": 3,**

**"from\_cache": False # ADD THIS**

**}**

**# Query each source with error handling**

**cosmos\_results = self.\_safe\_query\_cosmos(query, entities)**

**vector\_results = self.\_safe\_query\_vector(query)**

**vector\_ttl\_results = self.\_safe\_query\_vector\_ttl(query)**

**all\_results["data\_sources"] = {**

**"cosmos\_gremlin": {**

**"results": cosmos\_results,**

**"count": len(cosmos\_results),**

**"status": "success" if cosmos\_results else "no\_results"**

**},**

**"vector\_db": {**

**"results": vector\_results,**

**"count": len(vector\_results),**

**"status": "success" if vector\_results else "no\_results"**

**},**

**"vector\_db\_ttl": {**

**"results": vector\_ttl\_results,**

**"count": len(vector\_ttl\_results),**

**"status": "success" if vector\_ttl\_results else "no\_results"**

**}**

**}**

**# Phase 3: Generate enhanced context from all sources**

**self.\_populate\_enhanced\_context(all\_results, entities)**

**# ADD TO CACHE BEFORE RETURNING**

**self.\_add\_to\_cache(cache\_key, all\_results)**

**print(f"[IntegratedEntityAgent] Query complete: {len(entities)} entities, multiple data sources")**

**return all\_results**

**except Exception as e:**

**print(f"[IntegratedEntityAgent] Error processing query: {e}")**

**return {**

**"query": query,**

**"entities": [],**

**"context": [],**

**"text\_sections": [],**

**"relationships": [],**

**"confidence\_scores": {},**

**"overall\_confidence": 0.0,**

**"error": str(e),**

**"timestamp": datetime.now().isoformat(),**

**"extraction\_method": "integrated\_database\_enhanced\_error",**

**"total\_results": 0,**

**"from\_cache": False**

**}**

**Step 4: Add Cache Stats Endpoint**

**(Add after existing endpoints)**

**@app.route("/api/cache/stats", methods=["GET"])**

**def get\_cache\_stats():**

**"""Cache statistics"""**

**user = require\_auth()**

**if not user:**

**return jsonify({"error": "User not authenticated"}), 401**

**entity\_stats = orchestrator.entity\_agent.get\_cache\_stats()**

**return jsonify({**

**"entity\_cache": entity\_stats,**

**"performance\_gain": f"{entity\_stats['hit\_rate']}% queries cached",**

**"timestamp": datetime.now().isoformat()**

**})**

**@app.route("/api/cache/clear", methods=["POST"])**

**def clear\_cache():**

**"""Clear cache"""**

**user = require\_auth()**

**if not user:**

**return jsonify({"error": "User not authenticated"}), 401**

**orchestrator.entity\_agent.clear\_cache()**

**return jsonify({"message": "Cache cleared successfully"})**