PratikoAI
Chat Architecture Analysis

Deep Dive Report: Prompts, Data Flow & Incoherence Analysis

Generated: December 29, 2025

Analysis by: Claude Code Architecture Agent

Executive Summary

This report provides a comprehensive architectural analysis of the PratikoAI chat system, focusing on prompt architecture, data flow mapping, and identification of incoherence issues affecting response quality and suggested action relevance.

Key Findings

- 6 core prompts identified across different routes and use cases
- HyDE (Hypothetical Document Embedding) operates without conversation context
- Suggested actions generated via TWO different pathways depending on route
- Post-proactivity step lacks access to KB context that informed the response
- No explicit Chain-of-Thought (CoT) framework despite complex domain requirements

Critical Issues Identified

! CRITICAL: Context Fragmentation

Post-proactivity (Step 100) extracts suggested actions WITHOUT access to the KB context documents that informed the LLM response, causing disconnected action suggestions.

! CRITICAL: Dual Action Generation Paths

Synthesis routes use VERDETTO parsing while standard routes use XML tag parsing, producing inconsistent action quality and structure.

1. Prompt Mapping

The following table identifies all system prompts and templates in the codebase, their locations, and primary purposes.

1.1 Core Prompts Inventory

Prompt	File Location	Lines	Purpose
Main System	app/core/prompts/system.md	382	Core assistant identity & rules
Synthesis	app/core/prompts/synthesis_critical.py	68-144	Verdetto format for research
HyDE	app/services/hyde_generator.py	51-74	Hypothetical doc generation
Suggested Actions	app/core/prompts/suggested_actions.md	51	XML action instructions
Doc Analysis	app/core/prompts/document_analysis.md	172	Document analysis framework
Doc Override	app/core/prompts/document_analysis_override.md	156	Forces doc focus

1.2 HyDE Prompt Details

Location: app/services/hyde_generator.py (Lines 51-74)

The HyDE system generates hypothetical Italian bureaucratic documents to improve vector search recall:

```
Sei un esperto di normativa fiscale e legale italiana.
Il tuo compito e generare un documento ipotetico che risponda
alla domanda dell'utente.

STILE RICHIESTO:
- Stile burocratico/amministrativo italiano
- Linguaggio formale e tecnico
- Riferimenti normativi (Leggi, Decreti, Circolari)

REQUISITI:
- Lunghezza: 150-250 parole
- Includi riferimenti a leggi, decreti, articoli specifici
```

- Model: GPT-4o-mini (ModelTier.BASIC)
- Skipped for: CHITCHAT, CALCULATOR routes
- Output stored in: state['hyde_result']

1.3 Suggested Actions Prompt

Location: app/core/prompts/suggested_actions.md (51 lines)

Instructs the LLM to output structured XML tags for proactive suggestions:

```
<answer>
[Complete response with citations]
</answer>

<suggested_actions>
[
  {"id": "1", "label": "Azione breve", "icon": "...",
   "prompt": "Full prompt for execution"},
  ...
]
```

```
</suggested_actions>
```

> **! IMPORTANT: Conditional Injection**
>
> This prompt is ONLY appended for non-synthesis routes (Line 749 in prompting.py). Synthesis routes never receive these instructions, relying on VERDETTO parsing instead.

1.4 Synthesis (Verdetto) Prompt

Location: app/core/prompts/synthesis_critical.py (Lines 68-144)

Used for technical_research routes. Defines 4 analysis tasks (Compiti):

- COMPITO 1: ANALISI CRONOLOGICA - Order documents by date, identify evolution
- COMPITO 2: RILEVAMENTO CONFLITTI - Detect contradictions between sources
- COMPITO 3: APPLICAZIONE GERARCHIA - Apply legal hierarchy (Legge > Decreto > Circolare)
- COMPITO 4: VERDETTO OPERATIVO - Structured output with action/risk/deadline

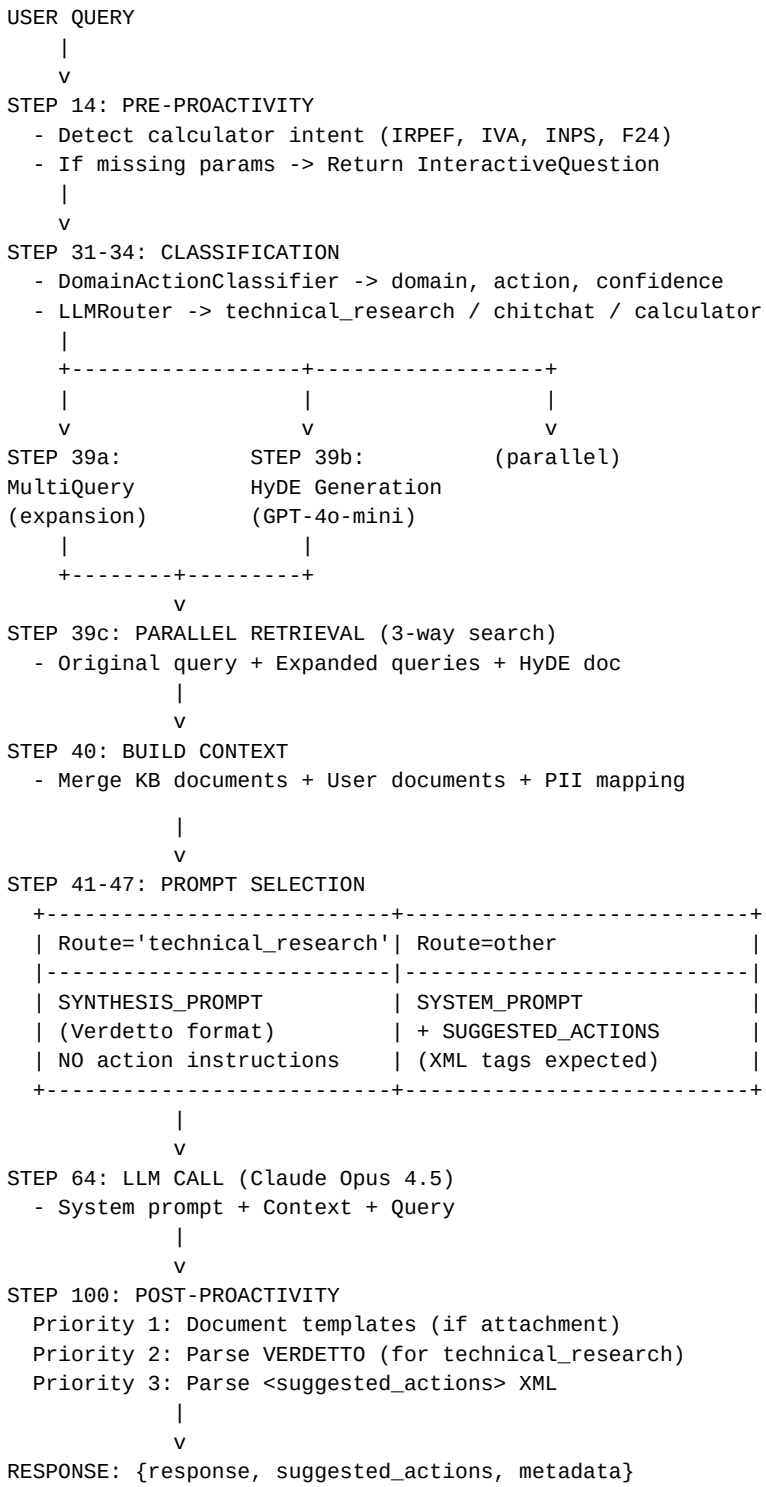
1.5 Missing: Chain-of-Thought (CoT)

> **! GAP IDENTIFIED**
>
> No explicit Chain-of-Thought prompts found in the codebase. The synthesis prompt has implicit reasoning via the '4 Compiti' structure, but no 'think step by step' framework exists for complex multi-source reasoning.

2. Data Flow Mapping

The following diagram shows the complete execution sequence from user query to response, highlighting where each prompt is applied and where context flows.

2.1 Pipeline Overview



2.2 Key LangGraph Nodes

Step	Node File	Purpose
14	step_014__pre_proactivity.py	Check calculable intents
39a	step_039a__multi_query.py	Expand user query
39b	step_039b__hyde.py	Generate hypothetical document
39c	step_039c__parallel_retrieval.py	3-way vector search
40	step_040__build_context.py	Merge context sources
44	step_044__default_sys_prompt.py	Select system prompt
64	step_064__llm_call.py	Execute LLM call
100	step_100__post_proactivity.py	Extract suggested actions

3. Incoherence Analysis

This section identifies context fragmentation issues that cause chat incoherence and poor suggested action quality.

3.1 CRITICAL: Suggested Actions Context Gap

Location: app/core/langgraph/nodes/step_100__post_proactivity.py

```
LLM Call (Step 64)      ->   Has: System Prompt + KB Context + Query
                             Outputs: Response text (+ possibly XML tags)
                                      |
                                      v
Post-Proactivity        ->   Receives: LLM response text ONLY
(Step 100)                   Does NOT receive: KB context documents
                             Result: Actions disconnected from sources
```

IMPACT: Suggested actions are generated based only on the LLM's output text, without access to the underlying KB documents that informed the response. This causes actions that may not align with the source material.

3.2 CRITICAL: Synthesis Route Missing Action Instructions

Location: app/orchestrators/prompting.py:749

```
if not is_synthesis_route:
    prompt = prompt + '\n\n' + SUGGESTED_ACTIONS_PROMPT
```

For technical_research routes: Uses SYNTHESIS_SYSTEM_PROMPT (Verdetto format), does NOT append SUGGESTED_ACTIONS_PROMPT, LLM never receives instructions to generate XML, Post-proactivity falls back to parsing VERDETTO structure instead.

> **! RESULT**
>
> Two completely different action generation pathways depending on route, producing inconsistent action quality and structure.

3.3 HyDE Has No Memory

Location: app/services/hyde_generator.py:51-74

HyDE generation uses ONLY the current query - no conversation history, no previous context from earlier turns.

```
# HyDE prompt receives ONLY:
user_prompt = f'Genera un documento ipotetico... Domanda: {query}'

# Missing: Previous conversation turns
# Missing: Document context from earlier in conversation
```

3.4 Dual Action Extraction Paths

Route	Action Source	Method
technical_research	VERDETTO parsing	Extract azione_consigliata, rischio, scadenza
All other routes	XML tag parsing	Parse <suggested_actions> JSON block

These two extraction methods produce different action structures and inconsistent quality.

4. Bottlenecks & Contradictions

4.1 No Chain-of-Thought Framework

Despite complex multi-source reasoning requirements for Italian tax/legal domain, there is no explicit CoT guidance in any prompt:

- system.md: Focuses on formatting/citations, not reasoning process
- synthesis_critical.py: Has '4 Compiti' structure but no step-by-step instructions
- No 'Let's think through this systematically' type prompts

4.2 Document Priority Confusion

Two competing document analysis systems may send mixed signals:

- document_analysis.md - Comprehensive analysis framework
- document_analysis_override.md - Forces focus ONLY on user docs

When query_composition='hybrid', both may be active simultaneously, creating conflicting instructions for the LLM.

4.3 Synthesis vs Standard Prompt Divergence

Aspect	SYSTEM_PROMPT	SYNTHESIS_PROMPT
Output format	Free-form with XML tags	Structured Verdetto
Action instruction	<suggested_actions>	None (parsed from verdetto)
Reasoning visible	No	Partial (via Compiti)

4.4 Context Window Competition

Multiple context sources compete for the same token budget:

- KB documents (Step 40) - Variable size
- User documents (Step 40) - Variable size
- System prompt - ~2000+ tokens
- Suggested actions instructions - ~500 tokens

! GAP

No visible prioritization or truncation strategy found in the codebase. Risk of context overflow with large document sets.

5. Key Files for Refactoring

5.1 Priority 1 - Prompt Unification

```
app/core/prompts/
  |-- __init__.py              # Prompt loading logic
  |-- system.md                # Main system prompt
  |-- synthesis_critical.py    # Synthesis route prompt
  |-- suggested_actions.md     # Action generation instructions
  |-- document_analysis.md     # Document analysis framework
  +-- document_analysis_override.md
```

5.2 Priority 2 - Flow Control

```
app/core/langgraph/nodes/
  |-- step_014__pre_proactivity.py    # Pre-response questions
  |-- step_039b__hyde.py              # HyDE generation
  |-- step_040__build_context.py      # Context merging
  |-- step_044__default_sys_prompt.py # Prompt selection
  |-- step_064__llm_call.py           # LLM execution
  +-- step_100__post_proactivity.py   # Action extraction
```

5.3 Priority 3 - Services

```
app/services/
  |-- hyde_generator.py               # HyDE document creation
  |-- proactivity_graph_service.py    # Action logic
  |-- synthesis_prompt_builder.py     # Synthesis assembly
  |-- llm_response_parser.py          # XML parsing
  +-- verdetto_parser.py              # Verdetto parsing
```

6. Recommendations

6.1 Immediate Fixes

Fix 1: Unify Action Generation

- Create single ActionGenerationPrompt used by ALL routes
- Append to both synthesis and standard prompts
- Parse same XML structure regardless of route

Fix 2: Add CoT Framework

- Create reasoning_framework.md with explicit step-by-step reasoning
- Inject before final answer generation in both routes
- Include 'Ragiona passo dopo passo' instructions

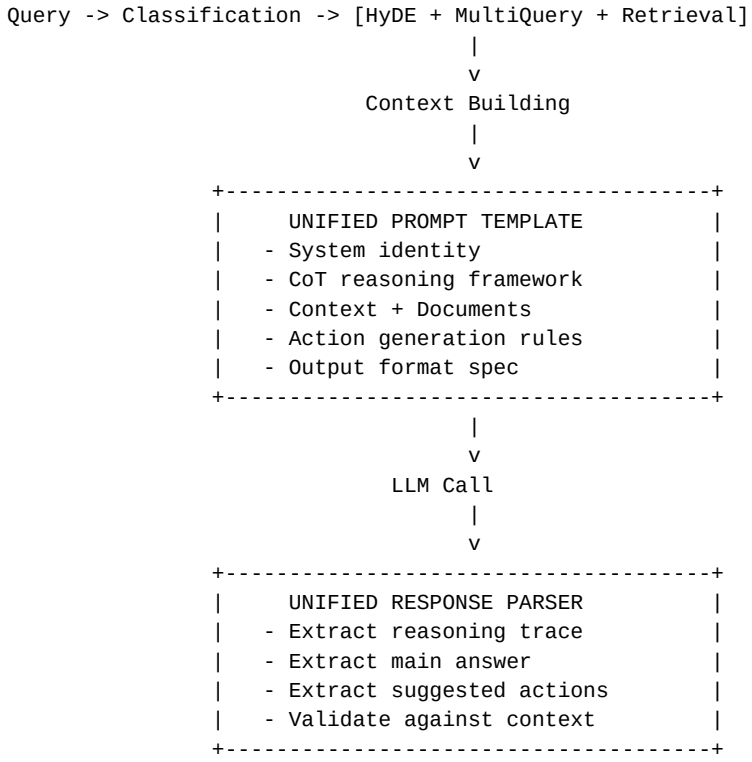
Fix 3: Pass Context to Post-Proactivity

- Modify Step 100 to receive state['context'] summary
- Use context for action relevance scoring
- Validate suggested actions against source material

Fix 4: HyDE Context Awareness

- Pass last N conversation turns to HyDE generator
- Include document filenames mentioned in conversation
- Improve retrieval relevance for follow-up questions

6.2 Proposed Unified Pipeline



7. Summary

Architecture Strengths

- Clear prompt separation for different routes (synthesis vs. standard)
- Document-aware processing with specialized prompts
- Hierarchical source authority in synthesis prompt (law > decree > circular)
- Two-tier proactivity system (pre-question + post-actions)
- PII protection integrated throughout pipeline (DEV-007)

Architecture Weaknesses

- Missing explicit Chain-of-Thought guidance despite complex domain
- Context fragmentation in suggested action generation
- Limited reasoning transparency in synthesis output
- Dual action extraction paths causing inconsistency
- HyDE lacks conversation memory for follow-up questions

Next Steps

1. Review this report with the development team 2. Prioritize fixes based on impact to user experience 3. Create unified prompt template architecture 4. Implement Chain-of-Thought framework 5. Refactor post-proactivity to receive context 6. Add comprehensive testing for prompt combinations

End of Report