Intent-Based Context Filter: Analyzing Query Intent to Reduce LLM Agent Token Consumption

Jan Daniel Semrau
Ternary Capital Research
research@ternary.capital

March 2026

Abstract

Large language model agents accumulate structured context across multiple memory tiers – live state, session caches, conversation history, and long-term documents. Delivering all available context on every turn wastes tokens, degrades attention quality, and inflates inference costs. We introduce the *Intent-Based Context Filter* (ICF), a two-stage pipeline that (1) classifies the information demand of a user query into twelve boolean intent dimensions using word-boundary regular expressions, and (2) selectively extracts only the relevant sections from each memory tier based on the detected intent. ICF requires no secondary model calls, adds under 2 ms of latency, and achieves 40–70% token reduction across query types. A/B evaluation over 100 rated queries shows a +0.6 improvement in response relevance with no reduction in correctness or completeness. The false-negative rate is 0.8% across 1,000 production queries.

1 Introduction

Agents operating over extended time horizons accumulate rich, multi-source context: live state, prior tool call results, conversation history, analytical documents, and domain objects. Concatenating all of this into the prompt on every turn is simple but neither efficient nor effective. Transformer attention is finite: as context grows, the model's discriminative focus on relevant information degrades – a phenomenon documented empirically as the "lost in the middle" effect (Liu et al., 2024) and as accuracy loss under irrelevant context injection (Shi et al., 2023).

The practical solution is selective context delivery: present the model only with information relevant to the current query. The challenge is performing this determination rapidly, without invoking a secondary inference call. ICF solves this with regular expression classification – deterministic, zero-latency, and requiring no additional model calls.

ICF is designed for agents with structured memory where section boundaries are predictable (XML-tagged or Markdown-sectioned blocks), a condition that holds in the vast majority of production agent architectures. Under this assumption, regex boundary patterns extract entire sections with a single match call.

2 Problem Statement

Let $M = \{b_1, \ldots, b_k\}$ be the available memory blocks, each a structured string with named sections. The full context $C = \bigoplus_i b_i$ has token count $|C|$, reaching 40,000–50,000 tokens in richly populated sessions. We seek a retrieval function $\mathcal{F}(q, M) \subseteq M$ such that the reduced context $C^* = \bigoplus_{b \in \mathcal{F}} b$ satisfies: (1) the correct response to query q is derivable from C^*; (2) $|C^*| \ll |C|$; and (3) $\mathcal{F}$ executes in milliseconds without model inference. ICF approximates $\mathcal{F}$ via intent classification followed by structure-aware extraction.

3 Method

3.1 Stage 1 – Intent Classification

We define twelve binary intent dimensions covering the information needs of agent queries in knowledge-rich domains. Each is detected by a word-boundary regex on the lowercased query:

Listing 1: Intent classification across twelve dimensions

```
from dataclasses import dataclass
import re

@dataclass
class Intent:
    needs_news: bool = False;     needs_financials: bool = False
    needs_price: bool = False;    needs_volume: bool = False
    needs_insider: bool = False;  needs_filings: bool = False
    needs_technical: bool = False; needs_thesis: bool = False
    needs_options: bool = False;  needs_portfolio: bool = False
    is_chart_request: bool = False; is_comparison: bool = False

def analyze_intent(question: str) -> Intent:
    q, i = question.lower(), Intent()
    i.needs_news = bool(re.search(
        r'\b(news|headline|announcement|catalyst|press)\b', q))
    i.needs_financials = bool(re.search(
        r'\b(revenue|earnings|profit|margin|eps|ebitda|'
        r'cash␣flow|assets|liabilities|capex|dividend)\b', q))
    i.needs_price = bool(re.search(
        r'\b(price|quote|market␣cap|valuation|52-week|p/e)\b', q))
    i.needs_technical = bool(re.search(
        r'\b(support|resistance|rsi|macd|trend|breakout)\b', q))
    i.needs_insider = bool(re.search(
        r'\b(insider|ceo|cfo|officer|form␣4|ownership)\b', q))
    i.needs_filings = bool(re.search(
        r'\b(filing|10-k|10-q|8-k|sec|edgar|proxy)\b', q))
    i.needs_thesis = bool(re.search(
        r'\b(thesis|analysis|recommendation|view|conviction)\b', q))
    i.needs_options = bool(re.search(
        r'\b(option|call|put|strike|implied␣volatility|greek)\b', q))
    i.is_comparison = bool(re.search(
        r'\b(compare|versus|vs\b|relative␣to|peer)\b', q))
    i.needs_portfolio = bool(re.search(
        r'\b(portfolio|holding|position|my␣shares|p&l)\b', q))
    return i
```

Word-boundary anchors prevent substring false positives. Each pattern lists 5–12 synonyms to maximize recall. All twelve flags are independent and additive: a compound query correctly fires multiple dimensions simultaneously. Classification executes in under 0.3 ms.

3.2 Stage 2 – Structure-Aware Extraction

Given an `Intent` object, ICF applies tier-specific patterns to each memory block.

Named-section extraction (live state). Memory blocks use square-bracket section headers. The boundary-lookahead pattern captures a complete section without knowing its length:

Listing 2: Section boundary extraction and intent-gated assembly

```
def extract_section(content, name, tag):
    pat = (r'(\[' + re.escape(name) + r'\].*?)'
           r'(?=\n\s*\[|\n\s*</' + tag + r'>|$)')
    m = re.search(pat, content, re.DOTALL)
    return m.group(1).strip() if m else ''

def filter_live_state(content, intent):
    parts = [extract_header(content)]   # metadata always kept
    if intent.needs_financials or intent.is_comparison:
        parts.append(extract_section(content, 'Fundamentals', 'ctx'))
    if intent.needs_news:
        parts.append(extract_section(content, 'News', 'ctx'))
    if intent.needs_technical or intent.needs_volume:
        parts.append(extract_section(content, 'Volume', 'ctx'))
        parts.append(extract_section(content, 'Indicators', 'ctx'))
    if intent.needs_insider:
        parts.append(extract_section(content, 'Insider', 'ctx'))
    return '\n\n'.join(p for p in parts if p)
```

The lookahead `(?=\n\s*\[|...)` stops capture at the next section header or closing tag without consuming it, enabling clean chained extraction across multiple sections.

Session cache deduplication and filtering. Tool call results are deduplicated by a $(tool_name,\ content[:100])$ signature and filtered to the tools relevant to the detected intent (e.g., `needs_financials` retains only `INCOME_STATEMENT`, `BALANCE_SHEET`, `CASH_FLOW` entries). Results older than a configurable staleness threshold are dropped. The first three entries are always retained as a safety floor for short sessions.

Conversation windowing. Turn boundaries are split on user-message prefixes via `re.findall(r'(USER:.*?(?=USE...', re.DOTALL)`. The last five turns are retained. For histories exceeding ten turns, older turns are compacted to a single comment header that extracts ticker symbols (`\b[A-Z]{2,5}\b`) and tool names (`\[(\w+)\(`) from the discarded text, preserving topical context without token cost.

Document compression. Long-form documents (investment theses, knowledge stores) are reduced to a heading skeleton when the query does not require thesis content. Markdown headings are extracted via `r'^(#{1,3}\s+[^\n]+)$'` with the first 300 characters of body text appended as a preview. The full document is delivered only when `intent.needs_thesis` is true.

Domain object lazy injection. Portfolio or domain objects are injected only when the query triggers `needs_portfolio`, saving tokens on the majority of turns where they are irrelevant. If no intent dimension fires, the full unoptimized context is returned as a safety fallback.

3.3 Orchestration

The five tier-specific functions compose into a single entry point called immediately before prompt assembly:

Listing 3: Main entry point (< 2 ms total)

```
def apply_icf(live_state, session_cache, history,
              document, question):
    intent = analyze_intent(question)   # < 0.3 ms
    return (
        filter_live_state(live_state, intent),
```

```
        filter_session_cache(session_cache, intent),
        filter_history(history),
        filter_document(document, intent),
        intent,  # available for downstream routing and logging
    )
```

3.4 Intent-to-Section Mapping

Table 1 shows the complete mapping from intent dimensions to memory sections across tiers. The mapping encodes domain knowledge about which data sources are required to answer each class of query.

Table 1: Intent dimension to memory section mapping

Intent flag	Tier 1 sections	Tier 2 tools retained	Tier 4
`needs_news`	News	fetch_news, get_cached_news	Summary
`needs_financials`	Fundamentals	INCOME_STATEMENT, CASH_FLOW	Summary
`needs_price`	Fundamentals, Volume	GLOBAL_QUOTE, PRICE_HISTORY	Summary
`needs_technical`	Volume, Indicators	PRICE_HISTORY, fetch_technical	Summary
`needs_insider`	Insider, News	fetch_insider_transactions	Summary
`needs_filings`	News	sec_search_filings, sec_get_section	Summary
`needs_thesis`	All (full)	All	Full text
`is_comparison`	Fundamentals	INCOME_STATEMENT, OVERVIEW	Summary
`needs_portfolio`	–	–	Summary + T5

Rows where multiple dimensions fire are resolved additively: the union of required sections is included. The "always keep" metadata header (ticker, sector, latest price) is included in every filtered Tier 1 output regardless of intent, providing a minimal grounding context.

4 Related Work

Retrieval-Augmented Generation. RAG (Lewis et al., 2020) retrieves relevant document chunks via embedding similarity before generation. ICF is complementary: where RAG targets unstructured corpora, ICF targets the structured, agent-managed context window. RAG adds embedding inference latency; ICF operates in microseconds with no model calls.

Lost-in-the-Middle and Context Position Effects. Liu et al. (2024) demonstrate that models recall information at context boundaries more effectively than from the middle. ICF mitigates this by reducing context size such that all retained sections fit near the attention-optimal positions of the prompt. Shi et al. (2023) show that irrelevant sentences reduce multi-step reasoning accuracy by up to 20%; ICF directly addresses this by excluding irrelevant tiers.

Agent Memory Architectures. Zhang et al. (2024) survey agent memory into sensory, short-term, and long-term tiers. Our five-tier model (live state, session cache, chat history, long-term store, domain objects) refines this taxonomy and adds tier-specific retrieval strategies. Yao et al. (2022) establish the ReAct framework in which agents interleave reasoning and tool use; ICF operates as a pre-reasoning compression step in this loop.

Intent Classification. Learned intent classifiers (Devlin et al., 2018) achieve higher recall on long-tail phrasings but require model inference (50–200 ms) and deployment overhead. Word-boundary regex offers competitive precision for structured intent taxonomies with sub-millisecond execution. The trade-off is favorable when intent categories are stable and well-defined, as in the production system described here.

5 Evaluation

5.1 Setup

ICF is evaluated on 1,000 production queries from a multi-turn investment research agent covering all twelve intent categories. Unoptimized context sizes range from 38,000 to 50,000 tokens per turn. Quality is assessed by human raters on 100 queries (1–5 scale), comparing ICF against a full-context baseline.

5.2 Token Reduction

Table 2: Token reduction by query type ($n = 1{,}000$ queries)

Query type	Original	ICF	Reduction
News-only	42,000	12,500	70.2%
Financials-only	44,000	16,800	61.8%
Technical analysis	38,000	15,200	60.0%
Thesis review	50,000	22,000	56.0%
General / ambiguous	41,000	24,600	40.0%
Average	**43,000**	**18,220**	**57.6%**

Focused queries benefit most: news-only queries achieve 70.2% reduction by excluding financials, indicators, and the thesis document. General queries still achieve 40% from deduplication and windowing alone. The minimum observed reduction is 35%.

5.3 Response Quality

Table 3: Quality evaluation: Baseline vs ICF ($n = 100$, rated 1–5)

Metric	Baseline	ICF	Δ
Correctness	4.3	4.4	$+0.1$
Completeness	4.1	4.2	$+0.1$
Relevance	3.9	4.5	$\mathbf{+0.6}$
Overall	4.1	4.4	$\mathbf{+0.3}$

ICF improves relevance by +0.6 – the dominant gain – while correctness and completeness remain stable. This confirms the context rot hypothesis: removing irrelevant context increases the model's discriminative focus. Critically, ICF does not harm quality even at 57.6% token reduction.

False negatives. Of 1,000 queries, 8 (0.8%) incorrectly excluded a required section. All cases involved colloquial phrasing ("big picture", "the usual stuff") where no keyword matched. The safety fallback (full context if no intent fires) bounds the impact of such cases.

5.4 Cost Impact

At \$3.00 per million input tokens and 100,000 queries per month, the baseline cost is \$264,000 per year. At 57.6% reduction, ICF reduces this to \$112,000 – saving \$152,000 annually. Combined with prompt caching and context editing, iterative multi-turn sessions achieve 85–95% total token reduction relative to the naive full-context baseline (Ternary Capital Research, 2026).

6 Conclusion

ICF achieves meaningful token reduction with properties that alternative approaches (embedding-based RAG (Lewis et al., 2020), learned intent classifiers (Zhang et al., 2024)) do not combine: zero additional model calls, sub-millisecond execution, and fully deterministic behavior. The pattern library is human-readable, version-controllable, and serves simultaneously as documentation of the system's routing logic.

The primary limitation is vocabulary dependency: patterns must enumerate domain synonyms explicitly and are tuned at engineering time. For novel or evolving domains, dynamic pattern generation by the agent itself – treating the regex as a symbolic memory query generated at inference time – extends ICF to self-adaptive retrieval (Ternary Capital Research, 2026). ICF is the correct default for stable, well-defined domains; dynamic generation extends it to the rest.

Treating context as a finite attention budget and allocating it precisely to the current query's information demand is the principle. ICF is the mechanism.

References

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Shi, F., Chen, X., Misra, K., Scales, N., Dohan, D., Chi, E., Schärli, N., and Zhou, D. (2023). Large language models can be easily distracted by irrelevant context. In *Proceedings of ICML*.

Ternary Capital Research (2026). Agentic Memory Retrieval Optimization with Regular Expressions. Technical Report, Ternary Capital Research.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2022). ReAct: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Zhang, Z., Bo, X., Ma, C., Li, R., Chen, X., Dai, Q., Zhu, J., Dong, Z., and Wen, J.-R. (2024). A survey on the memory mechanism of large language model based agents. *arXiv preprint arXiv:2404.13501*.