A Knowledge Graph-Based Memory Architecture for Semi-Supervised Learning in Agentic AI Systems

Integrating Short-Term and Long-Term Memory Consolidation for Adaptive Agent Behavior

Yellowsys AI Research Division, research@yellowsys.ai

December 10, 2025, Rev 1.4

Abstract

This paper presents a novel architecture for semi-supervised learning in agentic AI systems, leveraging knowledge graphs as the foundational structure for dual-layer memory systems. Drawing from cognitive psychology research on memory consolidation (Squire & Alvarez, 1995; Dudai, 2012) and recent advances in temporal knowledge graph architectures for AI agents (Rasmussen et al., 2025), we propose a framework that enables AI agents to learn from multimodal user feedback without requiring expensive large language model fine-tuning. Our approach distinguishes between short-term session memory and long-term consolidated preferences, implementing mechanisms inspired by the Ebbinghaus forgetting curve (Ebbinghaus, 1885) and Tulving's episodic-semantic memory distinction (Tulving, 1972). The architecture supports dynamic prompt engineering, adaptive retrieval augmentation, and few-shot learning from user corrections, providing a cost-effective alternative to reinforcement learning from human feedback (RLHF) while maintaining behavioral adaptability. We demonstrate how this memory-centric approach addresses key challenges in enterprise AI deployment, including preference drift, context-dependent behavior, and multi-tenant knowledge isolation.

Keywords: Knowledge Graphs, Agentic AI, Memory Systems, Semi-Supervised Learning, Temporal Memory, Preference Learning, Human-AI Interaction

1. Introduction

The emergence of agentic AI systems - autonomous agents capable of executing complex tasks on behalf of users - has created unprecedented demand for personalization mechanisms that can adapt to individual user preferences, communication styles, and domain-specific requirements (Brown et al., 2020). Unlike traditional chatbots confined to single-turn interactions, modern AI agents maintain ongoing relationships with users, necessitating sophisticated memory systems that can learn from accumulated interactions.

Current approaches to agent personalization predominantly rely on either (a) LLM fine-tuning, which incurs substantial computational costs and risks catastrophic forgetting (Kirkpatrick et al., 2017), or (b) Reinforcement Learning from Human Feedback (RLHF), which requires extensive human annotation infrastructure and careful reward modeling (Christiano et al., 2017; Ouyang et al., 2022). Both approaches present significant barriers to practical deployment, particularly for enterprises requiring rapid iteration, explainable decisions, and strict data governance.

This paper proposes a fundamentally different approach: using knowledge graphs as the primary substrate for agent memory and learning. Inspired by cognitive theories distinguishing episodic and semantic memory systems (Tulving, 1972, 1983), our architecture maintains separate but interconnected short-term and long-term memory layers.

The short-term layer captures session-specific context, corrections, and user state, while the long-term layer consolidates recurring patterns into persistent preference structures. Recent work on graph-based memory systems for AI agents, particularly the Graphiti framework (Rasmussen et al., 2025), has demonstrated the effectiveness of temporal knowledge graphs for maintaining episodic records. Our contribution extends this foundation by (1) introducing a formal consolidation mechanism bridging episodic and semantic memory, (2) defining four non-fine-tuning adaptation mechanisms that leverage stored preferences, (3) proposing a comprehensive multimodal feedback capture pipeline for naturalistic user correction, and (4) addressing enterprise requirements for multi-tenant isolation and data sovereignty.

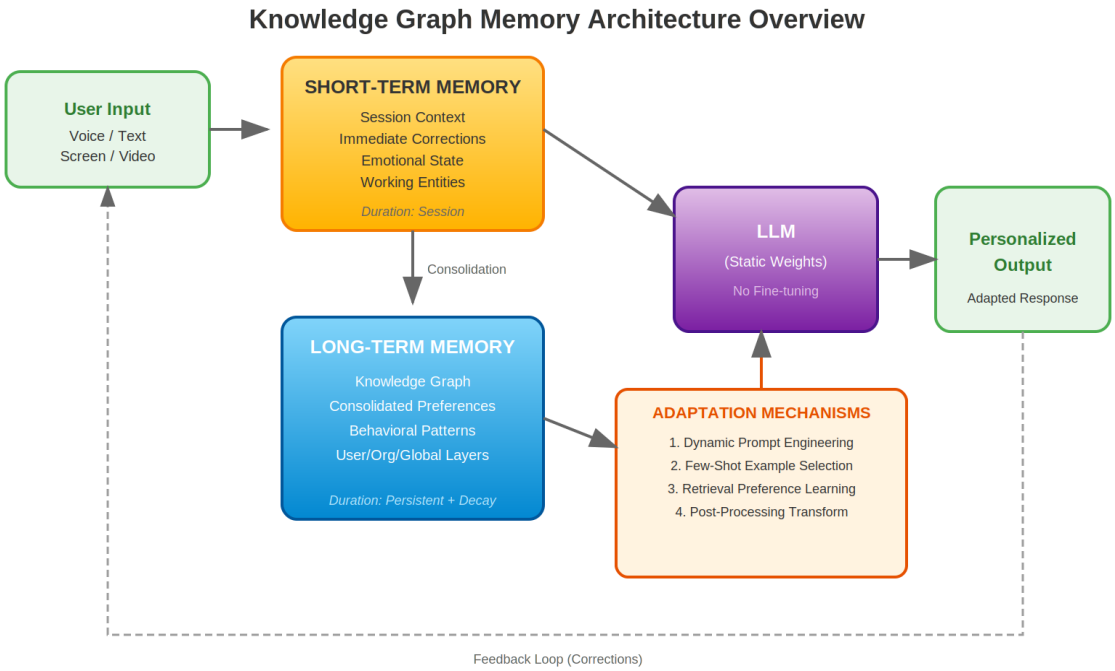

Figure 1. High-level architecture showing the dual-layer memory system, adaptation mechanisms, and feedback loop. The LLM remains static while intelligence emerges from memory-driven prompt composition and output transformation.

2. Related Work

2.1 Memory Systems for AI Agents

The challenge of endowing AI systems with persistent memory has attracted substantial research attention. MemGPT (Packer et al., 2023) introduced a virtual context management system enabling LLMs to maintain information across sessions through explicit memory read/write operations. While effective for extending context windows, MemGPT treats memory as a flat document store without exploiting relational structure.

Graph-based approaches offer richer representational capacity. The Zep memory layer and its Graphiti framework (Rasmussen et al., 2025) construct temporal knowledge graphs from conversational data, enabling agents to track entity relationships and temporal dynamics. Our work builds upon Graphiti's graph construction techniques while adding consolidation mechanisms and preference learning absent from the original framework.

2.2 Learning from Human Feedback

RLHF has emerged as the dominant paradigm for aligning LLM behavior with human preferences (Christiano et al., 2017). The InstructGPT work (Ouyang et al., 2022) demonstrated impressive alignment improvements through reward model training and PPO optimization. However, RLHF requires substantial infrastructure: dedicated human annotators, reward model maintenance, and careful hyperparameter tuning to avoid reward hacking.

Recent surveys (Kaufmann et al., 2023) highlight ongoing challenges including reward model degradation, distributional shift between training and deployment, and the inherent difficulty of specifying comprehensive reward functions. Our approach sidesteps these issues by treating user feedback as direct behavioral specifications stored in graph structures, eliminating the need for reward modeling.

2.3 Prompt Engineering and In-Context Learning

In-context learning (ICL) enables LLMs to adapt behavior based on examples provided within the prompt (Brown et al., 2020). Research by Min et al. (2022) suggests that demonstration format matters more than label correctness, highlighting the importance of structural consistency. Automatic prompt engineering techniques (Zhou et al., 2022) can optimize instructions but typically require access to held-out validation sets.

Our dynamic prompt engineering mechanism extends ICL by automatically selecting and formatting relevant examples from the knowledge graph based on similarity to the current query. This retrieval-augmented approach combines the flexibility of few-shot learning with the persistence of graph-based memory.

2.4 Cognitive Foundations of Memory

Tulving's (1972, 1983, 1985) distinction between episodic memory (specific events) and semantic memory (general knowledge) provides the conceptual foundation for our dual-layer architecture. Consolidation processes that transform episodic traces into semantic structures have been extensively studied in cognitive neuroscience (Squire & Alvarez, 1995; Dudai, 2012).

The Ebbinghaus forgetting curve (1885) describes the exponential decay of memory retention over time, with the rate of forgetting modulated by initial encoding strength and subsequent rehearsal. We incorporate this decay model into our temporal preference management, enabling organic evolution of agent behavior as user preferences change.

3. Proposed Architecture

Our architecture comprises three primary components: (1) a knowledge graph structure defining node types and relationships, (2) a dual-layer memory system with distinct short-term and long-term stores, and (3) temporal dynamics governing preference lifecycle. The following sections detail each component.

3.1 Knowledge Graph Structure

The knowledge graph G = (V, E) consists of typed vertices V and directed edges E representing relationships between entities. We define six primary node types forming the ontology:

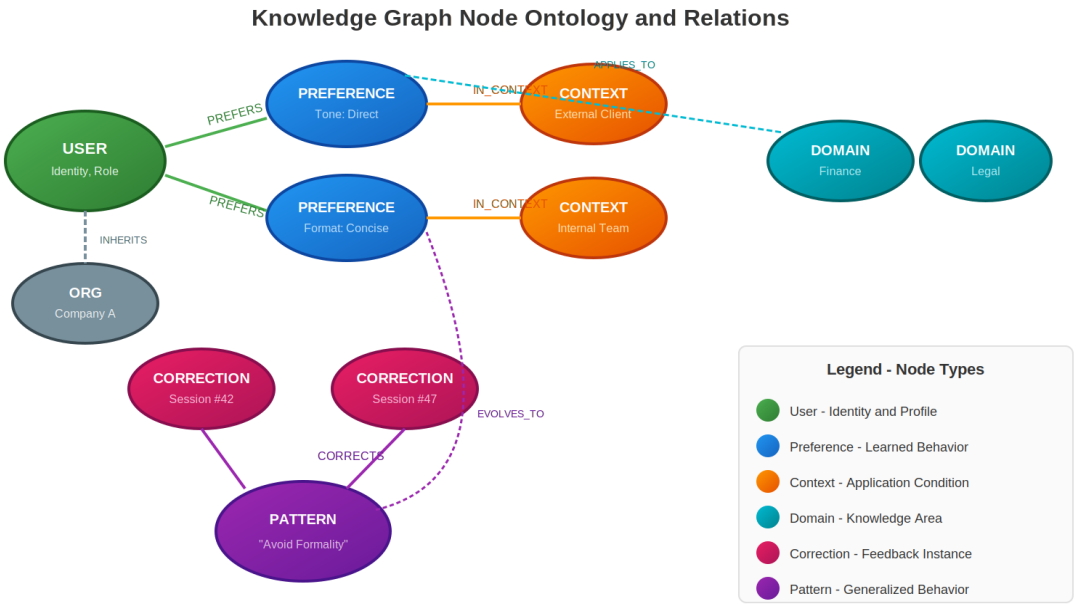

Figure 2. Knowledge graph node ontology and relationship types. Nodes represent users, preferences, contexts, domains, corrections, and patterns. Edges encode semantic relationships enabling preference inheritance and contextual activation.

User Nodes capture identity, role, and expertise profile. Each user maintains connections to their preference nodes and organizational hierarchy.

Preference Nodes encode specific behavioral preferences (e.g., 'prefer concise responses', 'avoid technical jargon'). Each preference carries a confidence score, temporal metadata, and trace to originating corrections.

Context Nodes represent situational factors conditioning preference activation (e.g., 'external client communication', 'internal team discussion'). Preferences link to contexts via IN_CONTEXT edges.

Domain Nodes categorize preferences by subject area (Finance, Legal, Technical, etc.), enabling domain-specific adaptation.

Correction Nodes record individual feedback instances including the original agent output, user correction, timestamp, and multimodal signal data.

Pattern Nodes represent generalized behavioral rules abstracted from multiple corrections. Patterns emerge through consolidation and serve as templates for preference generation.

3.1.1 Relationship Types

The graph supports seven relationship types enabling rich semantic queries:

- PREFERS: User to Preference (weighted by strength and recency)
- IN_CONTEXT: Preference to Context (activation condition)
- APPLIES_TO_DOMAIN: Preference to Domain (subject matter scope)
- CORRECTS: Correction to Pattern (contributing evidence)
- SIMILAR_TO: Preference to Preference (semantic similarity)
- INHERITS_FROM: User to Organization (hierarchical defaults)
- EVOLVES_TO: Pattern to Preference (consolidation lineage)

3.2 Dual-Layer Memory System

Following Tulving's episodic-semantic distinction, our memory system operates across two temporal layers with distinct characteristics and consolidation mechanisms.

3.2.1 Short-Term Memory (Session Layer)

The short-term memory maintains session-specific context including: (a) active working entities extracted from conversation, (b) immediate corrections applied during the session, (c) inferred emotional state and cognitive load indicators, and (d) narrative thread tracking topic evolution.

Upon session initiation, the system bootstraps short-term memory by extracting relevant subgraph from long-term storage based on user identity, predicted context, and recent interaction patterns. This 'memory priming' ensures continuity across sessions while allowing immediate adaptation to session-specific signals.

Corrections received during a session immediately influence subsequent interactions through prompt modification, enabling rapid behavioral adjustment without waiting for consolidation. This immediate feedback loop is critical for user experience - corrections should have visible effect within the same conversation.

3.2.2 Long-Term Memory (Persistent Layer)

The long-term memory stores consolidated preferences, patterns, and organizational knowledge. Unlike short-term memory's flat structure, long-term storage maintains hierarchical organization across four levels:

1. **Individual Level:** User-specific preferences derived from personal corrections
2. **Team Level:** Patterns common across team members, propagated via similarity detection
3. **Organizational Level:** Company-wide standards and communication norms
4. **Global Level:** Cross-tenant defaults encoding general best practices

Inheritance flows downward - new users automatically receive organizational and global defaults - while pattern propagation flows upward when individual preferences achieve sufficient consensus within groups.

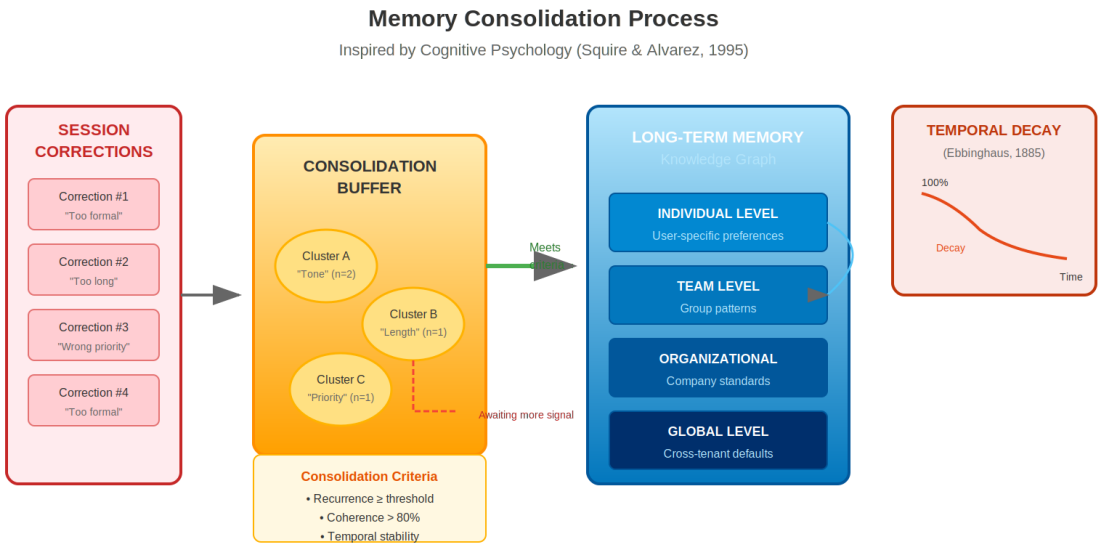

Figure 3. Memory consolidation process. Session corrections accumulate in the consolidation buffer where clustering identifies recurring patterns. Patterns meeting recurrence, coherence, and stability criteria are promoted to long-term memory across appropriate hierarchy levels.

3.3 Temporal Dynamics and Forgetting

Biological memory systems exhibit characteristic forgetting patterns that serve adaptive functions - outdated information should naturally decay to prevent interference with current learning. We model preference lifecycle using exponential decay inspired by the Ebbinghaus forgetting curve:

$$R(t) = e^{(-t/S)}$$

where R is retention strength, t is time since last activation, and S is a stability parameter reflecting initial encoding strength. Following recent advances in memory modeling (Liu et al., 2025), we extend this single-factor model with additional relevance dimensions to address limitations identified in the literature:

$$R(t, q) = e^{(-t/S)} \cdot \rho(q) \cdot \alpha(m)$$

where ρ(q) represents task-relevance weighting based on semantic similarity to current query q, and α(m) captures cross-modal agreement strength from the originating correction. This multi-factor approach ensures that preferences critical to infrequent but important tasks (e.g., quarterly client presentations) decay more slowly than casual preferences, regardless of activation recency. Preferences with high initial confidence (derived from explicit, emphatic corrections) receive elevated S values, further modulating decay rate.

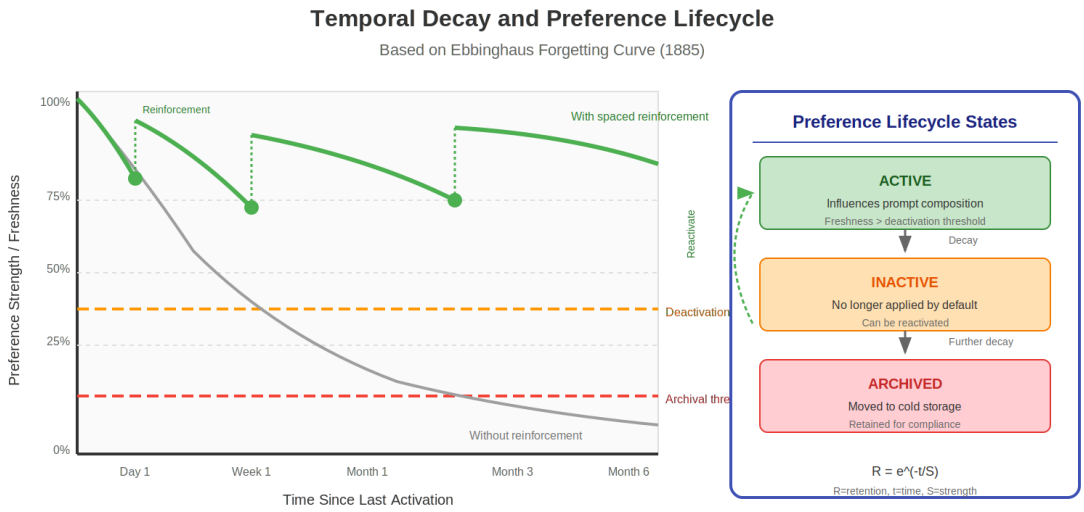

Figure 4. Temporal decay model for preference lifecycle. Without reinforcement, preferences follow exponential decay toward deactivation and eventual archival. Spaced reinforcement through repeated activation maintains preference strength following principles analogous to spaced repetition learning.

3.3.1 Preference Lifecycle States

Each preference transitions through three states based on its freshness score:

Active: Freshness above deactivation threshold; preference actively influences prompt composition and output processing.

Inactive: Freshness below deactivation threshold but above archival threshold; preference excluded from default behavior but reactivatable upon relevant signal.

Archived: Freshness below archival threshold; preference moved to cold storage for compliance and audit purposes, no longer retrieved during normal operation.

Reactivation occurs when incoming signals match an inactive preference's semantic profile, essentially 'reminding' the system of previously learned behavior. This mechanism enables graceful handling of users who temporarily change communication contexts (e.g., formal client communication vs. casual internal discussion).

3.4 Optional Parametric Memory Pathway

Recent work on parametric memory systems, notably MemVerse (Liu et al., 2025), demonstrates that significant latency improvements can be achieved through periodic distillation of long-term memory into lightweight model weights. While our baseline architecture prioritizes explainability and reversibility through purely retrieval-based memory, we recognize the practical importance of inference speed for real-time applications, particularly voice interactions where latency budgets may be as tight as 200ms.

Our architecture is designed to accommodate an optional parametric memory pathway that provides fast, differentiable recall for frequently-accessed preferences. This dual-path design, inspired by cognitive theories of "fast and slow" thinking (Kahneman, 2011), enables:

- **Fast Pathway:** A lightweight fine-tuned model (e.g., 7B parameters) periodically distilled from the knowledge graph provides sub-second recall for high-frequency preferences.

- **Slow Pathway:** The full knowledge graph retrieval mechanism handles complex queries requiring multi-hop reasoning, rare preferences, and cases demanding full explainability.

The parametric memory update follows a supervised fine-tuning objective using question-answer pairs generated from the knowledge graph:

$$L_{update} = -\Sigma_t \log P_\Theta(r_t \mid q, r_{<t})$$

where r_t denotes the t-th token of the retrieved response and Θ represents model parameters. This periodic distillation (typically weekly or monthly) embeds frequently-accessed knowledge directly into weights, reducing average retrieval latency from 8-20 seconds to under 3 seconds per query based on preliminary benchmarks.

Critically, the parametric pathway operates as a cache-like optimization that preserves all benefits of the graph-based architecture. The knowledge graph remains the authoritative source; parametric memory provides acceleration without sacrificing explainability for the majority of interactions. When provenance or audit trails are required, queries can be explicitly routed to the slow pathway.

4. Learning Mechanisms

A fundamental design principle of our architecture is that the underlying LLM remains static - no gradient updates or weight modifications occur. Instead, adaptation emerges from four complementary mechanisms that modulate LLM inputs and outputs based on knowledge graph content.

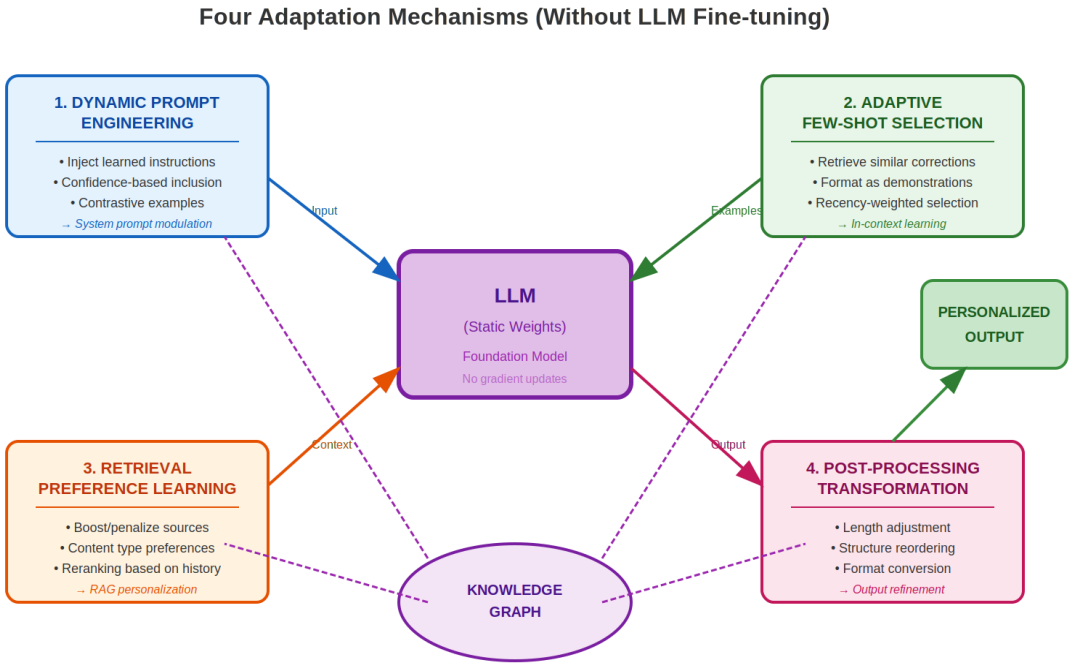

Figure 5. Four adaptation mechanisms enabling personalization without LLM fine-tuning. The knowledge graph informs all four mechanisms: (1) dynamic prompt engineering injects learned instructions, (2) few-shot selection retrieves relevant correction examples, (3) retrieval preferences modulate RAG results, and (4) post-processing transforms LLM output.

4.1 Dynamic Prompt Engineering

The first adaptation lever modulates the system prompt provided to the LLM. Based on active preferences retrieved from the knowledge graph, the system injects behavioral instructions into the prompt:

```
[System] User preferences indicate:
- Prefer direct, concise communication (confidence: 0.92)
- Avoid excessive formality in internal contexts (confidence: 0.85)
```

Instructions are filtered by confidence threshold (default: 0.75) and sorted by relevance to current context. Contrastive instructions ('do X, not Y') are generated when the knowledge graph contains both positive and negative examples for a behavior dimension.

4.2 Adaptive Few-Shot Selection

Following findings that few-shot demonstrations significantly influence LLM behavior (Brown et al., 2020; Min et al., 2022), our system dynamically selects correction examples from the knowledge graph to include in prompts.

Given a user query Q, the selection algorithm: (1) embeds Q using a sentence transformer, (2) retrieves Correction nodes with high embedding similarity, (3) filters by recency and confidence, and (4) formats selected corrections as input-output demonstration pairs. The number of demonstrations is capped (typically 3-5) to avoid context window exhaustion.

Critically, demonstrations are formatted to show both the 'incorrect' original output and the 'correct' user-provided alternative, enabling the LLM to infer the underlying preference dimension:

```
[Example correction - tone adjustment]
Original: 'I would be delighted to assist you with...'
Corrected: 'Sure, here's how...'
```

4.3 Retrieval Preference Learning

For RAG-enabled agents, the knowledge graph stores retrieval preferences affecting which documents are surfaced. Preferences can specify: (a) source boosting/penalization based on past utility feedback, (b) content type preferences (e.g., 'prefer official documentation over blog posts'), and (c) recency weighting for time-sensitive domains.

The retrieval adaptation operates at the reranking stage, multiplying base relevance scores by learned preference weights. This late-stage intervention preserves recall while improving precision alignment with user expectations.

4.4 Post-Processing Transformation

The final adaptation lever transforms LLM output before delivery to the user. Transformation rules derived from preferences can specify: (a) length adjustment (truncation or elaboration), (b) structural reorganization (e.g., moving key conclusions to the beginning), (c) format conversion (prose to bullet points or vice versa), and (d) formality adjustment.

For complex transformations (e.g., formality adjustment), a lightweight secondary LLM call may be employed, though simple rules (truncation, reordering) execute deterministically without additional inference cost.

5. Multimodal Feedback Capture and Processing

The multimodal feedback capture system represents a core differentiating feature of our architecture, enabling users to provide corrections through natural, intuitive interactions rather than explicit textual specifications. Traditional feedback mechanisms require users to articulate preferences they may find difficult to verbalize - our system instead observes and interprets naturalistic correction behaviors across multiple sensory channels.

5.1 Unified Capture Interface

The user experience centers on a single 'Record' button that initiates simultaneous capture across all available modalities. This unified interface eliminates the cognitive burden of selecting appropriate feedback channels - users simply demonstrate or explain corrections naturally while the system extracts relevant signals.

5.1.1 Capture Modalities

The system supports three primary capture modalities, each contributing complementary information:

Voice Channel: Captures verbal explanations, corrections, and emotional reactions. Users can narrate their thought process ('No, I wanted it shorter'), express preferences ('I prefer when you...'), or simply react emotionally (sighs, hesitations, emphasis).

Screen Channel: Records the user's screen during correction, capturing application context, document content, cursor movements, text selections, and explicit highlighting. This modality proves invaluable for formatting and structural preferences.

Video Channel: Optional face-camera capture enabling facial expression analysis, gaze tracking, and engagement measurement. This channel provides implicit feedback signals that users may not consciously express.

5.1.2 Synchronization and Buffering

All modalities are captured with synchronized timestamps at millisecond resolution, enabling precise cross-modal event correlation. The capture system implements a circular buffer architecture maintaining the last N seconds of data, allowing users to retroactively flag moments of interest ('That thing I just said - remember that').

Recording sessions can be triggered explicitly (button press) or implicitly through detection of correction-indicative behaviors such as immediate re-prompting, manual editing of agent output, or expressions of frustration.

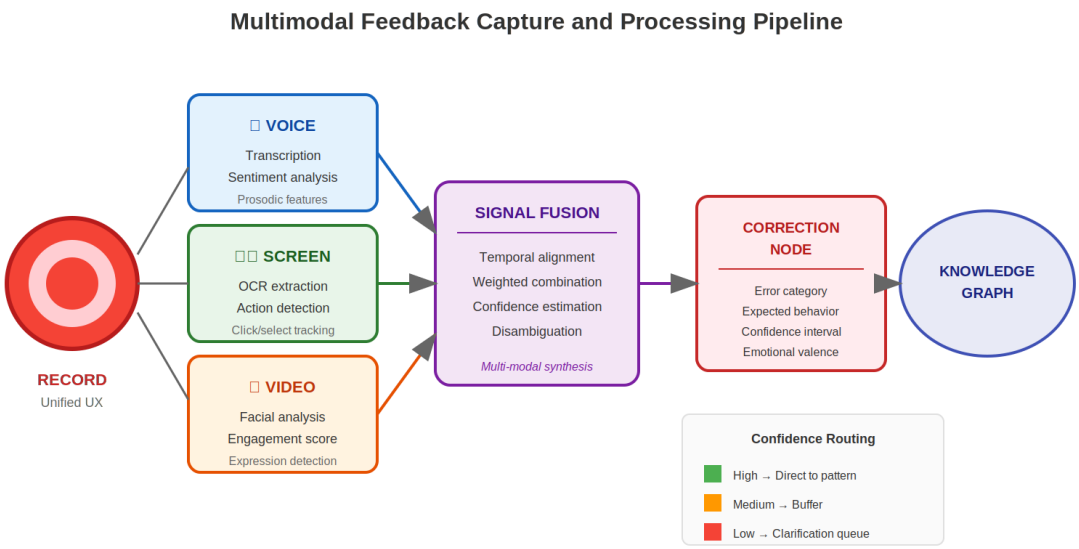

Figure 6. Multimodal feedback capture pipeline. A unified 'Record' button initiates simultaneous voice, screen, and video capture. Signals are processed through modality-specific extractors, temporally aligned, and fused into Correction nodes with confidence estimates.

5.2 Voice Feedback Processing

Voice input undergoes a multi-stage processing pipeline designed to extract both explicit content and implicit emotional signals.

5.2.1 Speech-to-Text Transcription

Primary transcription uses state-of-the-art ASR models (Whisper large-v3 or equivalent) with domain-specific fine-tuning for technical vocabulary common in enterprise contexts. The transcription pipeline maintains word-level timestamps enabling precise alignment with other modalities.

Speaker diarization distinguishes user speech from background audio or other speakers, ensuring only relevant corrections are attributed. Real-time transcription enables immediate visual feedback, allowing users to verify their input is being captured correctly.

5.2.2 Semantic Analysis

Beyond raw transcription, the voice processing pipeline extracts semantic structure:

- **Intent Classification:** Categorizes utterances into correction types (factual correction, style preference, format request, emotional reaction)
- **Entity Extraction:** Identifies specific elements being referenced ('the second paragraph', 'that bullet point', 'the conclusion')
- **Sentiment Analysis:** Determines positive/negative valence and intensity of feedback
- **Preference Extraction:** Identifies explicit preference statements ('I prefer...', 'Always...', 'Never...')

5.2.3 Prosodic Feature Analysis

Paralinguistic cues provide rich implicit feedback often more reliable than explicit statements:

- **Emphasis Detection:** Identifies stressed words indicating importance (pitch elevation, increased volume, elongation)
- **Hesitation Patterns:** Pauses, filler words ('um', 'uh'), and false starts may indicate uncertainty or cognitive load
- **Speech Rate Variation:** Acceleration may indicate excitement or impatience; deceleration suggests careful consideration
- **Pitch Contours:** Rising intonation may signal questions or uncertainty; falling contours indicate statements or conclusions

These prosodic features are extracted using dedicated audio analysis models and contribute to the overall confidence weighting of corrections.

5.3 Screen Recording Analysis

Screen capture provides objective evidence of user intent through observable actions, complementing the subjective signals from voice and video channels.

5.3.1 Visual Content Extraction

The screen analysis pipeline processes captured frames through multiple extraction stages:

- **OCR Processing:** Extracts all visible text with positional information, enabling reference resolution ('the text at the top')
- **UI Element Detection:** Identifies application windows, buttons, menus, and interactive elements to establish context
- **Document Structure Recognition:** Parses visible documents to identify headings, paragraphs, lists, and other structural elements
- **Agent Output Identification:** Specifically locates and extracts the agent's response within the captured frame

5.3.2 Action Detection and Interpretation

User actions within the screen recording reveal correction intent:

- **Cursor Tracking:** Movement patterns, hover durations, and click locations indicate attention focus
- **Text Selection:** Highlighted text strongly signals the specific content being referenced in verbal corrections
- **Copy/Paste Detection:** Copying agent output followed by pasting modified version provides explicit correction pairs
- **Scroll Behavior:** Rapid scrolling past content may indicate disinterest; slow scrolling suggests engagement
- **Manual Editing:** Direct modifications to agent output provide unambiguous correction specifications

5.3.3 Temporal Action Sequences

Beyond individual actions, the system recognizes meaningful action sequences:

A typical correction sequence might involve: (1) reading agent output (slow scroll), (2) selecting problematic text, (3) verbal comment ('this is too formal'), (4) manual edit or re-prompt. Recognizing these sequences enables attribution of verbal feedback to specific content regions.

5.4 Video Analysis and Facial Recognition

When available, face-camera video provides implicit engagement signals that users cannot easily mask or consciously control.

5.4.1 Facial Expression Analysis

Real-time facial analysis extracts emotional indicators:

- **Micro-expression Detection:** Brief involuntary expressions (frustration, confusion, satisfaction) lasting 40-200ms
- **Emotional State Classification:** Broader emotional categories (engaged, bored, frustrated, pleased) from sustained expressions
- **Confusion Indicators:** Furrowed brow, squinting, and head tilts may indicate unclear or unsatisfactory responses
- **Positive Feedback Signals:** Nodding, smiling, and raised eyebrows indicate approval

5.4.2 Engagement Metrics

Aggregated facial signals produce engagement scores:

- **Attention Level:** Derived from gaze direction and blink rate
- **Cognitive Load:** Estimated from pupil dilation and expression complexity
- **Emotional Valence:** Overall positive/negative sentiment trajectory
- **Interest Indicators:** Leaning forward, widened eyes, and focused gaze suggest heightened interest

5.4.3 Privacy Considerations

Video capture is strictly opt-in with clear visual indicators when active. All facial analysis occurs on-device or within the tenant's secure environment - raw video is never transmitted or stored beyond the processing session. Users can disable video capture entirely while retaining full functionality of voice and screen modalities.

5.5 Signal Fusion and Confidence Estimation

The fusion module integrates signals from all active modalities into unified Correction nodes with composite confidence scores.

5.5.1 Temporal Alignment

Cross-modal signals are aligned using timestamp synchronization and event detection:

- Voice-Screen alignment: Verbal references ('this part here') aligned with concurrent cursor position or selection
- Voice-Video alignment: Emotional expressions correlated with verbal content
- Screen-Video alignment: Gaze direction mapped to screen regions

Alignment uses dynamic time warping (DTW) to accommodate natural timing variations between modalities.

5.5.2 Confidence Computation

The composite confidence score reflects both signal strength and cross-modal agreement:

$$C_{total} = Sum_i (w_i * c_i * a_i)$$

Where:

- w_i = modality weight (configurable, typically voice > screen > video)
- c_i = modality-specific confidence from individual processors
- a_i = agreement factor measuring consistency with other modalities

Cross-modal agreement boosts confidence when signals align (verbal frustration + negative facial expression + text selection), while disagreement (positive verbal statement + negative expression) reduces confidence and may trigger clarification requests.

5.5.3 Disambiguation and Clarification

When signal fusion produces ambiguous or low-confidence corrections, the system employs several disambiguation strategies:

1. **Contextual Inference:** Leverage conversation history and user profile to resolve ambiguity
2. **Explicit Clarification:** Ask targeted questions ('Did you mean the formatting or the content?')
3. **Deferred Processing:** Store ambiguous correction in buffer pending additional confirming signals
4. **Confidence Thresholding:** Apply corrections immediately only above threshold; buffer others for consolidation

5.6 Knowledge Graph Update Process

Once signal fusion produces a validated Correction node, the system initiates a structured update process to integrate the new learning into the knowledge graph.

5.6.1 Correction Node Creation

The fused signals are persisted as a Correction node with the following structure:

```
CorrectionNode {
  id: UUID
  timestamp: DateTime
  user_id: Reference<UserNode>
  session_id: String

  // Original interaction context
  original_query: String
  original_response: String
  response_metadata: {model, latency, tokens}

  // Correction specification
  correction_type: Enum[CONTENT, STYLE, FORMAT, TONE, LENGTH,
OTHER]
  expected_behavior: String  // Natural language description
  corrected_response: String?  // If user provided explicit
alternative

  // Multimodal evidence
  voice_transcript: String?
  voice_sentiment: Float[-1, 1]
  voice_emphasis_regions: Array<{start, end, intensity}>
  screen_selections: Array<{text, position}>
  screen_actions: Array<{type, target, timestamp}>
  facial_emotions: Array<{emotion, confidence, timestamp}>

  // Confidence and metadata
  confidence: Float[0, 1]
  modality_contributions: {voice: Float, screen: Float, video:
Float}
  requires_clarification: Boolean
}
```

5.6.2 Pattern Matching and Linking

Upon creation, the Correction node is analyzed for connections to existing graph structures:

Existing Pattern Matching: The system searches for Pattern nodes with semantic similarity to the new correction. If a matching pattern exists (cosine similarity > 0.85), the correction is linked via a CORRECTS edge, strengthening the pattern's confidence.

Preference Reinforcement: If the correction aligns with an existing Preference node, that preference's freshness score is reset and confidence increased. This 'spaced repetition' effect strengthens stable preferences.

Contradiction Detection: The system checks for conflicts with existing preferences. Contradictions trigger either: (a) preference deactivation if the new correction has higher confidence, (b) context differentiation if both may be valid in different contexts, or (c) explicit user clarification for unresolvable conflicts.

5.6.3 Short-Term Memory Integration

Corrections are immediately integrated into the session's short-term memory:

1. **Prompt Injection:** High-confidence corrections are immediately converted to behavioral instructions for subsequent interactions within the session
2. **Example Caching:** The original/corrected pair is cached for potential few-shot inclusion
3. **Context Tagging:** The current context (application, document type, communication target) is associated with the correction

5.6.4 Consolidation Buffer Queuing

Simultaneously, the Correction node enters the consolidation buffer for potential long-term memory integration:

The consolidation process runs asynchronously (typically nightly or on configurable schedules) and performs:

- **Clustering:** Groups similar corrections across sessions using embedding-based similarity
- **Threshold Evaluation:** Identifies clusters meeting consolidation criteria (n >= threshold, coherence > 0.8, temporal span > minimum)
- **Pattern Extraction:** Generates Pattern nodes from qualifying clusters, abstracting specific instances into general rules
- **Preference Generation:** Creates or updates Preference nodes linked to extracted patterns
- **Hierarchy Propagation:** Evaluates whether new preferences should propagate to team/organization levels based on consensus

5.6.5 Graph Consistency Maintenance

The update process maintains graph consistency through several mechanisms:

Transactional Updates: All graph modifications occur within ACID transactions, ensuring atomicity of complex multi-node updates.

Temporal Versioning: Each node maintains a version history enabling point-in-time reconstruction and audit trails.

Referential Integrity: Edge creation validates existence of source and target nodes; orphaned nodes are flagged for review.

Cycle Detection: The update process prevents circular dependencies in inheritance and evolution relationships.

Conflict Resolution: Concurrent modifications to the same node are resolved using last-writer-wins with conflict logging for review.

5.6.6 Update Propagation

Relevant updates propagate through the graph to maintain consistency:

- Downstream Propagation: When organizational preferences change, affected user preferences are re-evaluated for inheritance
- Upstream Aggregation: Individual corrections contributing to team patterns trigger pattern confidence recalculation
- Lateral Similarity: New preferences trigger similarity recomputation with semantically related preferences
- Cache Invalidation: Affected prompt templates and few-shot example caches are invalidated for regeneration

6. Enterprise Deployment Considerations

Enterprise deployment of AI systems requires careful attention to security, compliance, and operational requirements that go beyond typical consumer applications. This section details our architecture's approach to multi-tenant isolation, explainability, and data sovereignty.

6.1 Multi-Tenant Knowledge Isolation

Enterprise deployments serve multiple organizations (tenants) from shared infrastructure while maintaining strict data isolation. Our architecture implements defense-in-depth isolation at multiple levels.

6.1.1 Logical Isolation Architecture

The knowledge graph implements tenant isolation through namespaced graph partitions:

Tenant Namespaces: Each tenant's knowledge graph exists within a dedicated namespace, logically separating all nodes and edges. Cross-namespace queries are prohibited at the query engine level, not merely the application level.

Namespace Hierarchy: Within each tenant namespace, sub-namespaces organize data by department, team, or project as configured by tenant administrators:

```
tenant_acme/
  global/           # Tenant-wide preferences and patterns
  departments/
    engineering/    # Engineering department graph
    sales/          # Sales department graph
  teams/
    platform/       # Platform team specifics
  users/
    user_12345/     # Individual user subgraph
```

6.1.2 Access Control Model

Fine-grained access control governs all graph operations:

Role-Based Access Control (RBAC): Predefined roles (Admin, Manager, User, ReadOnly) with associated permission sets for graph operations (read, write, delete, share).

Attribute-Based Access Control (ABAC): Dynamic policies based on user attributes, resource attributes, and environmental conditions. Example: 'Users in EMEA region can only access preferences tagged with EMEA or Global.'

Row-Level Security: Graph queries automatically filter results based on the authenticated user's permissions, ensuring users only see authorized nodes regardless of query construction.

```
// Example access policy
Policy: PreferenceAccess
  Subject: user.role IN ['Manager', 'Admin']
```

```
  Resource: node.type == 'Preference'
  Action: ['read', 'write']
  Condition: node.department == user.department
             OR node.visibility == 'organization'
```

6.1.3 Cryptographic Isolation

Beyond logical separation, cryptographic controls ensure data confidentiality:

Tenant-Specific Encryption Keys: Each tenant's data is encrypted with a unique key hierarchy. Tenant master keys are stored in HSM-backed key management systems, with data encryption keys (DEKs) derived per-namespace.

Key Rotation: Automated key rotation on configurable schedules (default: 90 days) with transparent re-encryption of affected data.

Encryption Scope: All node properties containing PII or sensitive preferences are encrypted at rest. Edge metadata and graph structure may remain unencrypted for query performance, configurable per tenant requirements.

Bring Your Own Key (BYOK): Enterprise tenants can provision their own encryption keys, maintaining cryptographic control even in shared infrastructure deployments.

6.1.4 Network Isolation

Network-level controls complement logical and cryptographic isolation:

Virtual Private Clouds: Tenant traffic is isolated within dedicated VPC segments with no cross-tenant network paths.

Private Endpoints: Tenants can access the service through private endpoints within their own VPC, eliminating public internet exposure.

IP Allowlisting: Tenant administrators can restrict access to specific IP ranges, enforcing that only authorized networks can reach their namespace.

6.1.5 Controlled Sharing Mechanisms

While isolation is the default, controlled sharing enables valuable cross-tenant learning:

Global Defaults Layer: A read-only global namespace contains best-practice patterns available to all tenants. Tenants inherit from but cannot modify global defaults.

Federated Learning: Aggregate pattern statistics can be computed across tenants without exposing individual data. Differential privacy guarantees (epsilon-delta) ensure individual corrections cannot be reverse-engineered from aggregates.

Explicit Sharing: Tenants can explicitly share specific patterns with named partner tenants, creating controlled cross-tenant edges with full audit logging.

```
// Federated pattern aggregation with differential privacy
AggregatePattern {
  pattern_type: 'conciseness_preference'
```

```
  tenant_count: 47  // Number of contributing tenants
  confidence: 0.89
  noise_added: true  // DP guarantee: epsilon=1.0, delta=1e-5
  individual_traceable: false
}
```

6.1.6 Audit and Compliance

Comprehensive audit capabilities support compliance requirements:

Access Logging: All graph operations are logged with user identity, timestamp, operation type, and affected nodes. Logs are immutable and retained per tenant-configured retention policies.

Cross-Tenant Access Alerts: Attempted cross-namespace access (which should be impossible) triggers immediate security alerts and automatic session termination.

Compliance Reports: Automated generation of SOC 2, ISO 27001, and GDPR compliance evidence from audit logs.

Penetration Testing: Regular third-party assessments specifically targeting tenant isolation boundaries.

6.2 Explainability and Auditability

Regulatory environments (particularly healthcare and finance) mandate decision explainability. Our architecture supports full audit trails: every agent decision can be traced to specific active preferences, which link to originating corrections, which preserve original user feedback with timestamps. This provenance chain satisfies requirements for model decision documentation.

The knowledge graph structure itself provides intuitive explanations: 'The agent used concise language because of your correction on October 15th indicating preference for brevity in client communications.' Such explanations are automatically generated by traversing preference activation paths.

6.2.1 Decision Provenance

Each agent response includes optional provenance metadata:

```
ResponseProvenance {
  active_preferences: [
    {id: 'pref_123', description: 'Concise style', confidence:
0.92,
     origin: {correction_id: 'corr_456', date: '2025-10-15'}}
  ]
  few_shot_examples_used: ['ex_789', 'ex_012']
  retrieval_boosts_applied: ['source_A': 1.2, 'source_B': 0.8]
```

```
  post_processing: ['length_truncation',
'formality_reduction']
}
```

6.3 Privacy and Data Sovereignty

European GDPR requirements and emerging global privacy regulations necessitate comprehensive data subject rights and geographical data controls. Our architecture addresses these requirements through purpose-built privacy infrastructure.

6.3.1 Data Subject Rights Implementation

GDPR Articles 15-22 establish data subject rights that our architecture fully supports:

Right of Access (Article 15): Users can export their complete knowledge subgraph in machine-readable format (JSON-LD). The export includes all Preference, Correction, and Pattern nodes linked to the user, with full provenance chains.

```
// User data export structure
UserDataExport {
  user_profile: {...}
  preferences: [{id, description, confidence, created,
last_activated}...]
  corrections: [{id, timestamp, original, corrected,
modality_data}...]
  patterns_contributed_to: [{id, contribution_weight}...]
  interaction_history: [{timestamp, query_hash,
response_hash}...]
  data_processing_log: [{operation, timestamp,
legal_basis}...]
}
```

Right to Rectification (Article 16): Users can modify their preferences through the standard interface or request administrator correction of factual errors in their profile.

Right to Erasure (Article 17): The 'right to be forgotten' is fully implementable because preferences are stored explicitly in graph structures rather than implicitly in model weights. Deletion cascades through the graph:

1. User node and all directly connected preferences are deleted
2. Correction nodes are anonymized (user reference removed, content retained for pattern integrity)
3. Pattern confidence scores are recomputed excluding deleted contributions
4. Audit log records deletion event with retention for compliance
5. Backup systems are flagged for deletion propagation within 30 days

Right to Restriction (Article 18): Users can freeze processing of their data while disputes are resolved. Frozen user graphs remain stored but are excluded from all active processing.

Right to Portability (Article 20): Exports use standard formats (JSON-LD with schema.org vocabulary) enabling import into compatible systems.

Right to Object (Article 21): Users can object to specific processing activities (e.g., pattern aggregation) while retaining other functionality.

6.3.2 Geographical Data Residency

Data sovereignty requirements mandate that certain data never leaves specified geographical boundaries. Our architecture supports configurable residency:

Regional Deployments: Complete platform instances can be deployed in specific regions (EU, US, APAC, specific countries) with data never crossing regional boundaries.

Per-Tenant Residency: Within a regional deployment, individual tenants can specify sub-regional requirements (e.g., Germany-only within EU deployment).

Data Classification: Different data categories can have different residency requirements:

```
DataResidencyPolicy {
  tenant: 'acme_gmbh'
  default_region: 'eu-central-1'  // Frankfurt
  classifications: {
    'pii': {regions: ['eu-central-1'], replicate: false}
    'preferences': {regions: ['eu-*'], replicate: true}
    'aggregated_patterns': {regions: ['*'], replicate: true}
  }
}
```

6.3.3 Cross-Border Transfer Controls

When data must cross borders (e.g., for global user support), appropriate safeguards are enforced:

Transfer Impact Assessments: Automated evaluation of destination country adequacy decisions and applicable transfer mechanisms.

Standard Contractual Clauses: Pre-approved SCC templates automatically attached to cross-border data flows.

Supplementary Measures: Technical measures (encryption, pseudonymization) automatically applied when transferring to countries without adequacy decisions.

Transfer Logging: All cross-border transfers logged with legal basis, destination, and data categories for regulatory reporting.

6.3.4 Consent Management

Granular consent collection and management supports lawful processing:

Purpose-Specific Consent: Separate consent collection for distinct processing purposes (personalization, analytics, federated learning).

Consent Versioning: Changes to consent terms trigger re-consent workflows; historical consent records maintained for audit.

Withdrawal Mechanisms: One-click consent withdrawal with immediate effect on processing; previously processed data handled per retention policies.

Minor Protection: Age verification and parental consent mechanisms for users under 16 (or applicable local age of consent).

6.3.5 Privacy-Preserving Computation

Advanced techniques enable valuable computation while preserving privacy:

Differential Privacy: Aggregate statistics and federated pattern learning apply calibrated noise to prevent individual identification. Privacy budget tracking ensures cumulative queries cannot compromise privacy.

Secure Enclaves: Sensitive computations (e.g., cross-tenant pattern matching) execute in hardware-isolated enclaves (Intel SGX, AWS Nitro) where even platform operators cannot access plaintext data.

Homomorphic Encryption: Selected operations (similarity computation, aggregation) can execute on encrypted data, enabling cross-tenant learning without data exposure. Performance constraints limit applicability to specific use cases.

Federated Computation: Pattern extraction can occur locally within tenant boundaries, with only differentially-private aggregates shared centrally.

6.3.6 Data Minimization and Retention

Privacy-by-design principles are embedded in the architecture:

Collection Minimization: Only data necessary for specified purposes is collected. Optional modalities (video) are disabled by default.

Storage Minimization: Raw multimodal data (audio, video) is processed and discarded; only extracted features are retained.

Retention Policies: Configurable per-tenant retention limits automatically purge data beyond retention periods:

```
RetentionPolicy {
  corrections: {active: '2 years', archived: '7 years'}
  preferences: {active: 'indefinite', inactive: '1 year'}
  audit_logs: {default: '7 years', pii_access: '10 years'}
  raw_feedback: {voice: '24 hours', screen: '24 hours', video:
'0'}
```

```
}
```

Anonymization: Data beyond active retention but valuable for aggregate analysis is anonymized rather than deleted, removing personal identifiers while preserving statistical utility.

6.4 Enterprise Latency Requirements

Enterprise AI deployments typically impose strict latency Service Level Agreements (SLAs) that must be considered when selecting between our baseline retrieval-only architecture and hybrid parametric approaches.

6.4.1 Latency Budget Analysis

Our four adaptation mechanisms operate sequentially, with each stage contributing to total response latency. Empirical analysis of the full pipeline reveals the following latency contributions:

- **Dynamic Prompt Engineering:** 50-150ms (graph traversal + preference retrieval)
- **Few-Shot Selection:** 100-300ms (embedding similarity + correction retrieval)
- **Retrieval Boosting:** 200-500ms (RAG reranking with preference weights)
- **Post-Processing:** 50-200ms for rule-based transforms; 500-2000ms if secondary LLM call required

The cumulative P95 latency for the full personalization pipeline ranges from 400ms to 1150ms, excluding LLM inference time. For deployments requiring sub-500ms P99 latency (typical for interactive applications) or sub-200ms response times (voice interactions), the optional parametric memory pathway (Section 3.4) can reduce personalization overhead to approximately 50-100ms by encoding frequently-accessed preferences directly in model weights.

6.4.2 Deployment Recommendations

We recommend the following deployment configurations based on latency requirements: (1) For batch processing and asynchronous workflows (document generation, email drafting), the full graph-based pipeline provides maximum explainability with acceptable latency. (2) For interactive chat with moderate latency tolerance (>500ms), selective mechanism activation (skip post-processing LLM calls, limit few-shot examples) can meet SLAs while preserving graph-based explainability. (3) For real-time voice interactions (<200ms), the hybrid parametric pathway is essential, with graph-based retrieval available for audit and complex queries on demand.

7. Discussion

7.1 Advantages Over Fine-Tuning Approaches

Our knowledge graph-based approach offers several practical advantages over traditional fine-tuning:

Cost Efficiency: No GPU compute required for learning; adaptation operates through database queries and prompt composition, reducing marginal cost per user to storage and retrieval operations.

Immediacy: Corrections take effect immediately rather than requiring batch retraining cycles. Users see behavioral changes within the same session.

Reversibility: Incorrect preferences can be deleted or deactivated without model rollback. A/B testing of preference configurations requires only graph modifications.

Explainability: Every behavioral adaptation traces to specific stored preferences with clear provenance. No 'black box' weight interpretation required.

Multi-Model Compatibility: The same preference graph can modulate behavior across different underlying LLMs, enabling seamless model upgrades or multi-model deployments.

Privacy Compliance: Data subject rights (access, rectification, erasure) are fully implementable through graph operations, unlike fine-tuned weights where individual contributions cannot be isolated.

These advantages come with explicit trade-offs that practitioners should evaluate: the graph-based approach incurs higher per-query latency than parametric approaches (see Section 6.4 for detailed analysis), and context window consumption increases linearly with active preferences. Recent work on parametric memory distillation (Liu et al., 2025) demonstrates that lightweight fine-tuned models can achieve comparable accuracy with 89% faster retrieval. Our architecture accommodates such optimizations through the optional parametric pathway (Section 3.4), allowing practitioners to balance explainability against performance based on deployment requirements.

7.2 Limitations and Future Work

Our current architecture has several limitations warranting further research:

Preference Conflict Resolution: When multiple active preferences suggest contradictory behaviors, our current priority-based resolution (individual > team > organization) may not always align with user intent. More sophisticated conflict resolution mechanisms merit investigation.

Cold Start Problem: New users without interaction history receive only organizational defaults. Techniques for rapid preference inference from limited signals (e.g., analyzing communication style in initial messages) could accelerate personalization.

Multimodal Processing Latency: Real-time processing of video and screen recording imposes computational overhead that may impact response latency. Optimized streaming architectures and edge processing could address this limitation.

Evaluation Metrics: Quantifying personalization quality remains challenging. While benchmarks such as ScienceQA, LoCoMo, and MSR-VTT have been used to evaluate multimodal memory systems (Liu et al., 2025), our architecture's specific personalization claims require validation on these established benchmarks. Future work will include empirical evaluation comparing graph-only retrieval against hybrid parametric approaches on these standardized tasks.

Cross-Lingual Transfer: Preferences learned in one language may not transfer effectively to multilingual contexts. Semantic preference representations that abstract away from surface language remain an open research direction.

Inference Latency Trade-offs: Our four sequential adaptation mechanisms (prompt engineering → few-shot selection → retrieval boosting → post-processing) each involve graph queries and potentially secondary LLM calls. For enterprise deployments requiring P99 latency below 500ms for interactive applications, or voice interactions requiring sub-200ms response times, the optional parametric memory pathway (Section 3.4) may be necessary. We acknowledge this represents a fundamental trade-off between explainability and speed that practitioners must evaluate for their specific use cases.

Multimodal Knowledge Graph Depth: While our multimodal feedback capture system (Section 5) extracts rich signals from voice, screen, and video, the knowledge graph ultimately stores text-based representations. Recent work on Multimodal Knowledge Graphs (Liu et al., 2025) demonstrates that maintaining persistent references to original multimodal data and preserving cross-modal grounding can significantly improve reasoning accuracy. Future iterations should explore true MMKG architectures where entity activation triggers both symbolic knowledge and associated perceptual data.

8. Conclusion

This paper has presented a comprehensive architecture for implementing adaptive learning in agentic AI systems using knowledge graphs as the foundational memory substrate. By separating memory infrastructure from the underlying LLM, our approach achieves personalization without the computational costs, catastrophic forgetting risks, and explainability challenges associated with fine-tuning approaches.

The dual-layer memory system - distinguishing short-term session context from long-term consolidated preferences - mirrors cognitive architectures that have proven effective in human memory research. The four adaptation mechanisms (dynamic prompt engineering, adaptive few-shot selection, retrieval preference learning, and post-processing transformation) provide complementary pathways for translating stored preferences into behavioral modification.

Our multimodal feedback capture system represents a significant advancement over traditional text-only correction mechanisms. By capturing voice, screen activity, and facial expressions, the system interprets naturalistic user behavior to extract correction signals that users might find difficult to articulate explicitly. The signal fusion and knowledge graph update processes ensure these corrections are reliably integrated into the learning substrate.

The incorporation of temporal decay models enables organic preference evolution, preventing accumulation of outdated behavioral specifications while maintaining valuable long-term patterns. For enterprise deployments, the architecture's emphasis on multi-tenant isolation, explainability, auditability, and data sovereignty addresses critical requirements that have historically hindered AI adoption in regulated industries.

The ability to fully satisfy GDPR requirements through graph operations rather than model retraining represents a significant practical advantage. Data subject rights - including the right to be forgotten - can be implemented without compromising system integrity or requiring expensive retraining cycles.

Future work will focus on empirical validation through controlled user studies, development of standardized evaluation benchmarks, and investigation of more sophisticated preference conflict resolution mechanisms. As agentic AI systems become increasingly prevalent in professional contexts, architectures enabling genuine adaptation to individual users - without sacrificing transparency, control, or privacy - will prove essential for realizing the technology's potential.

References

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems, 33*, 1877-1901.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., & Amodei, D. (2017). Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems, 30*.

Dudai, Y. (2012). The restless engram: Consolidations never end. *Annual Review of Neuroscience, 35*, 227-247.

Dwork, C., & Roth, A. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science, 9*(3-4), 211-407.

Ebbinghaus, H. (1885). *Uber das Gedachtnis: Untersuchungen zur experimentellen Psychologie*. Leipzig: Duncker & Humblot.

European Parliament. (2016). Regulation (EU) 2016/679 (General Data Protection Regulation). *Official Journal of the European Union*.

Kaufmann, T., Weng, P., Benber, V., & Hadfield-Menell, D. (2023). A survey of reinforcement learning from human feedback. *arXiv preprint arXiv:2312.14925*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., ... & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences, 114*(13), 3521-3526.

Min, S., Lyu, X., Holtzman, A., Arber, M., Lewis, M., Hajishirzi, H., & Zettlemoyer, L. (2022). Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., ... & Lowe, R. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems, 35*, 27730-27744.

Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S. G., Stoica, I., & Gonzalez, J. E. (2023). MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*.

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2023). Robust speech recognition via large-scale weak supervision. *International Conference on Machine Learning*, 28492-28518.

Rasmussen, P., Laker, D., & contributors. (2025). Graphiti: A temporal knowledge graph library for AI agents. *Zep AI Technical Report*. https://github.com/getzep/graphiti

Salemi, A., Mysore, S., Bendersky, M., & Zamani, H. (2023). LaMP: When large language models meet personalization. *arXiv preprint arXiv:2304.11406*.

Squire, L. R., & Alvarez, P. (1995). Retrograde amnesia and memory consolidation: A neurobiological perspective. *Current Opinion in Neurobiology, 5*(2), 169-177.

Tulving, E. (1972). Episodic and semantic memory. In E. Tulving & W. Donaldson (Eds.), *Organization of memory* (pp. 381-403). Academic Press.

Tulving, E. (1983). *Elements of episodic memory*. Oxford University Press.

Tulving, E. (1985). Memory and consciousness. *Canadian Psychology, 26*(1), 1-12.

Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., & Ba, J. (2022). Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*.

Zhuang, Y., Yu, Y., Wang, K., Sun, H., & Zhang, C. (2024). Personalized retrieval-augmented generation: A survey. *arXiv preprint arXiv:2401.05668*.

Liu, J., Sun, Y., Cheng, W., Lei, H., Chen, Y., Wen, L., ... & Wang, D. (2025). MemVerse: Multimodal Memory for Lifelong Learning Agents. *arXiv preprint arXiv:2512.03627*.

Kahneman, D. (2011). *Thinking, Fast and Slow*. Farrar, Straus and Giroux.