Long-Running Agent with Memory & Context Management

Transform the B2B Lead Discovery Agent into a long-running autonomous agent with persistent memory, state management, checkpointing, and parallel execution capabilities.

Current State Analysis

The current implementation is a single-run agent with:
- Sequential 5-stage pipeline (discovery -> structure -> role -> enrichment -> verification)
- Basic JSON storage for leads only
- No persistent memory across sessions
- No checkpoint/resume capability
- No failure recovery mechanism
- No context engineering or selective retrieval

Proposed Architecture

The new architecture includes:
1. Controller (Workflow) - Handles planning, state management, and checkpointing
2. Memory System - Three types: Short-term, Long-term, and Working memory
3. Execution Loop - Plan -> Act -> Observe -> Remember -> Re-Plan cycle
4. Auto-recovery on failures with retry logic

Proposed Changes - New Files

[NEW] memory/memory_manager.py

Core memory management with three memory types:
- Short-term: Current session context
- Long-term: Persistent storage for past leads and learnings
- Working memory: Temporary scratchpad for reasoning

[NEW] memory/state_manager.py

State persistence and checkpointing:
- Serialize pipeline state at each stage
- Resume from last checkpoint
- Track execution history
- Handle failures with retry logic

[NEW] memory/context_builder.py

Intelligent context engineering:
- Build optimized context per agent
- Retrieve relevant past experiences
- Summarize large contexts
- Inject learnings from past successes/failures

Proposed Changes - Modified Files

[MODIFY] workflow.py

- Add checkpoint creation after each stage
- Add resume capability from any checkpoint
- Add auto-retry on failure with re-planning

- Integrate memory system for context injection

[MODIFY] database.py

Add new collections:
- memory.json - Long-term memory storage
- checkpoints.json - Pipeline state checkpoints
- execution_history.json - Action logs

[MODIFY] agent.py

New commands:
- resume: Resume interrupted analysis
- status: Check running/paused analyses
- forget: Clear specific memory
- learn: View what agent has learned

[MODIFY] agents/*.py

- Accept memory context as input
- Use past analyses for better decisions
- Log decisions to memory for future reference

Verification Plan

1. Checkpoint/Resume Test

- Start analysis, interrupt with Ctrl+C during enrichment stage
- Resume with 'resume' command
- Should continue from enrichment stage

2. Memory Persistence Test

- Session 1: Analyze a company (e.g., Google)
- Quit and restart
- Session 2: Analyze related company (Alphabet)
- Should show context from previous Google analysis

3. Failure Recovery Test

- Temporarily disable API key to cause failure
- Run analysis - should checkpoint and pause
- Re-enable API key and resume
- Should continue from failure point

Implementation Order

1. Create memory/ module with memory manager and state manager
2. Update database.py with new collections
3. Update workflow.py with checkpointing
4. Update individual agents to use memory
5. Update agent.py with new commands
6. Manual testing and verification