🧠 Агенты с памятью: Memory Stores и Dreaming

🧠 Агенты с памятью: Memory Stores и Dreaming

Ключевые тезисы:

- 🎯 **Проблема**: Агенты по умолчанию изолированы и не помнят информацию между сессиями.
- 💾 **Решение 1: Memory Store** — постоянное файловое хранилище, которое можно подключать к сессиям для чтения и записи.
- ✨ **Решение 2: Dreaming** — асинхронный процесс, который организует, обогащает и проверяет информацию в Memory Store.
- 🔗 **Композиция**: Сессии + Memory Store + Dreaming создают мощную систему для долгосрочной работы агентов.

🚫 Базовый случай: Изолированные агенты

В текущей реализации агенты и их сессии работают изолированно:

- Информация, переданная в одной сессии, недоступна в другой.
- **Пример**: Сообщили агенту в сессии А о докладе "CMA". При запросе в сессии Б агент не знает об этой информации.

💾 Memory Store: Файловая память для агентов

Memory Store — это постоянное файловое хранилище, похожее на файловую систему, которое подключается как ресурс к сессиям.

🎯 Ключевые возможности:

- **Персистентность**: Данные сохраняются между сессиями.
- **Гибкость**: Можно создавать множество хранилищ (на пользователя, рабочую область и т.д.).
- **Мощный интерфейс**: Монтируется как файловая система. Агент может использовать `bash`, `grep`, читать и записывать файлы.
- **Контроль доступа**: Можно настроить доступ только для чтения (`read-only`) или для чтения и записи (`read-write`).

🔧 **Как использовать:**

1. Создать Memory Store через CLI или Console.
2. Подключить его к сессии, указав `memory_store_id` и опциональный `prompt` для фокусировки агента.
3. Агент автоматически получает инструменты для работы с хранилищем.

📊 **Пример работы:**

- **Сессия 1 (Запись)**: Агенту сообщают новую информацию. Он записывает её в файл внутри Memory Store.
- **Сессия 2 (Чтение)**: Новый агент в другой сессии, подключённый к тому же хранилищу, ищет информацию с помощью `grep` и находит ранее записанные данные.

✨ Dreaming: Улучшение памяти асинхронно

Проблема: Агенты могут бесконечно "сваливать" информацию в Memory Store, что приводит к его разрастанию, беспорядку и устареванию данных.

Dreaming — это фоновый асинхронный процесс (задача), который анализирует и улучшает Memory Store.

🎯 **Что делает Dreaming:**

- ✅ **Проверяет факты** (fact-checking).
- 🔄 **Организует и консолидирует** информацию.
- 🗑️ **Удаляет дубликаты** (deduplication).

- 📝 **Обогащает данные**: добавляет недостающие детали, даты, метаданные.
- 🗂️ **Создаёт индекс** для эффективного поиска.

🔧 **Как запустить:**

1. Указать входной Memory Store и список ID сессий (транскриптов) для анализа.
2. Выбрать модель (например, Claude Opus для качества или Sonnet для экономии).
3. Добавить опциональные инструкции для кастомизации.
4. Запустить задачу. Она работает асинхронно, от нескольких минут до часов.

⚙️ **Принцип работы под капотом:**

- Это **мульти-агентская система**.
- **Оркестратор** запускает **суб-агентов** (по одному на каждую входную сессию) для exhaustive-анализа.
- Процесс **недеструктивный**: создаётся новый **выходной (output) Memory Store**, входной не изменяется.
- Обеспечивается **полная наблюдаемость**: можно смотреть на сессию самого Dreaming-процесса в Console.

📈 **Результат**: Выходное хранилище содержит лучше организованную, проверенную и обогащённую информацию, что повышает эффективность и "интеллект" будущих агентов.

🖥️ Управление через Console и CLI

- **Console** предоставляет UI для просмотра Memory Stores (как файловый менеджер), Dreaming-задач и их диффов.
- **CLI** позволяет управлять всем программно: создавать хранилища, запускать dreaming, проверять статусы.
- **Человек в цикле**: Можно вручную редактировать файлы в Memory Store или проверять результаты Dreaming перед использованием.

💡 Композиция и лучшие практики

1. **Сессия** → Изолированный инстанс агента (эпиhemeral).
2. **+ Memory Store** → Возможность переноса информации между сессиями.
3. **+ Dreaming** → Поддержание памяти в актуальном, организованном и богатом состоянии при масштабировании.

⚖️ **Вопрос стоимости (токенов):**

- Dreaming использует много токенов, так как стремится к exhaustive-анализу.
- **Высокий cache hit-rate** (~95%) благодаря агентской обработке.
- **Способы контроля**: выбор более лёгкой модели, настройка промпта, бюджетирование. В будущем возможны скидки по аналогии с Batch API.

🎯 Выводы

- **Память — ключ** к преодолению изоляции агентов.
- **Memory Store** решает проблему переноса данных, но создаёт вызов управления этой памятью.
- **Dreaming** — это системный подход к поддержанию памяти в оптимальном состоянии с помощью других агентов.
- Вместе эти функции позволяют строить сложные, долгоживущие и интеллектуальные агентские workflows.
