AI researcher

Problem Framing & Assumptions

The goal of this project is to design an AI research agent that can take an open-ended question and produce a structured, evidence-backed report. The focus is not on producing a single best answer, but on demonstrating how an agent can plan, gather information, and synthesize results using external tools.

Overview

In this project, I have implemented a tool-using AI research agent designed to transform an open-ended user prompt into a structured report. This system focuses on grounded reasoning, where the agent does not rely solely on internal model knowledge, but actively searches for and retrieves external sources and analyzes them before generating conclusions.

Given a natural language prompt, the agent is able to perform this process:

1. Planning
    A planning LLM takes in the user's request into a set of research tasks, which represents different investigation angles.

2. Research loop
    For each task, the agent is able to use a web tool to locate relevant sources, retrieves the webpage content, and then extracts structured claims and supporting evidence using an LLM extractor.

3. Evidence Aggregation
    This extracted information is stored in a structured evidence store, where traceability between claims and sources is preserved.

4. Report Generation
    A final LLM call is able to synthesize the evidence into a human-readable Markdown research report.

This design separates each step, which allows the agent to operate as a modular research system rather than a single monolithic model call.

User Prompt
↓
Planner LLM
↓
Research Loop Controller
├── Search Tool (Tavily)
├── Page Retriever

```
├── Evidence Extractor LLM
↓
Structured Evidence Store
↓
Report Writer LLM
↓
Markdown Research Report
```

Agent Loop Design

```
The core of this system is the iterative search loop, alternating between
reasoning and tool use. Rather than producing a report in a single model call, the
agent follows the following cycle:

1. Task Decomposition
    The planner LLM converts a high-level question into smaller research tasks.
This reduces reasoning complexity and encourages coverage across multiple
investigation dimensions.
2. Tool-Driven Exploration
    For each task, the agent interacts with external tools:
        - A web search API provides candidate sources
        - A retriever fetches raw webpage content
        This ensures that conclusions are grounded in external information rather
than purely model-generated knowledge.
3. Evidence Extraction
    An LLM-based extractor converts unstructured webpage texts into structured
evidence objects. Each object contains:
        - Claim
        - Supporting Evidence
        - Classification (risk or benefit)
        - Confidence Score
        - Source URL
    This structured format enables traceability and reduces the risk of
unsupported conclusions.
4. Stateful Evidence Accumulation
    Extracted evidence is stored in a central evidence store which allows the
agent to:
        - Avoid repeating work
        - Track source diversity
        - Build a cumulative knowledge base across iterations
5. Synthesis Phase
    Once sufficient evidence has been gathered, a report-writer LLM synthesizes
the structured knowledge into a human-readable Markdown report.

This loop demonstrates a reasoning->action->observation->memory pattern.
```

Data & Trace Model

```
The system uses structured intermediate representations rather than free-form text
to maintain transparency and traceability.

Each piece of extracted evidence follows this schema:
{
claim: string,
evidence: string,
type: "risk" | "benefit",
confidence: float,
source_url: string
}
```

Observability & Metrics

To evaluate the performance of the research agent, the following metrics are useful:

Metric	Purpose
Evidence objects per task	Measures research depth and coverage
Source diversity	Reduces bias and over-reliance on single domains
Retrieval success rate	Monitors reliability of web access and tools
Extraction confidence average	Tracks quality and certainty of extracted evidence
Iterations per question	Controls API usage and computational cost

Reliability & Safety Considerations

```
During testing, the agent encountered real-world issues such as:
    - HTTP  errors from websites blocking automated scraping
    - Pages with low-quality or irrelevant content
    - Potential hallucination during extraction

To address this, the system includes:
    - Domain filtering to avoid known blocked sites
    - Multiple sources per task to improve robustness
    - Structured evidence storage to reduce unsupported claims
    - Confidence scoring to signal extraction uncertainty

These measures demonstrated the need for resilience when building tool-using
agents in open web environments.
```

Tradeoffs

- Depth vs. Cost: More iterations increase coverage but raise API usage - Simplicity vs. Autonomy: A fixed loop is stable while a self-reflecting agent is more powerful but complex - Scraping vs API retrieval:

Scraping increases coverage but reduces reliability

Future System

```
With more time, the system could evolve into:
    - A multi-agent architecture
    - Citation validation and fact-checking modules
    - A long-term knowledge graph memory
    - Source credibility scoring
    - Asynchronous task execution
```

Proof-of-Concept Implementation

```
The provided code implements a minimal working agent with:
 - A planning LLM
 - A web search tool
 - A webpage retriever
 - An LLM-based extractor
 - A report-writer module
```

This design demonstrates how a tool-using agent can perform grounded, multi-step research while maintaining traceability, modularity, and robustness in real-world web environments.