# ResearchMCP – AI-Powered Research Contradiction & Consensus Detector

## Overview

ResearchMCP is an MCP (Model Context Protocol) powered research assistant that helps users analyze academic papers not just by summarizing them, but by detecting where research findings agree, contradict, or remain uncertain.  
  
It’s an agentic RAG system built on top of Claude (as the reasoning LLM) and FastMCP (as the tool orchestration layer). Instead of simply pulling text from a vector store, it reasons about the retrieved content — identifying patterns, contradictions, and consensus across multiple papers.  
  
The system integrates open-access academic APIs (like OpenAlex and arXiv) to fetch papers dynamically, processes their abstracts, and uses the LLM’s reasoning power to compare claims and summarize the research landscape on any topic.

## Problem Statement

Researchers, students, and data scientists struggle with information overload. Thousands of new papers are published weekly, especially in fast-evolving fields like AI, ML, and NLP. While summarization tools and chatbots can condense single papers, they fail to compare multiple studies or reveal when research findings conflict.  
  
Existing RAG systems just retrieve text and summarize it — they do not analyze relationships between findings. ResearchMCP solves this by introducing a multi-step, agentic reasoning chain that evaluates papers’ claims, finds contradictions, and highlights research gaps or consensus.

## Project Goals

1. Retrieve the most relevant papers from OpenAlex/arXiv for a given topic or question.  
2. Extract the main claims or findings from each paper (via LLM reasoning).  
3. Compare the claims across papers to detect agreements, contradictions, and uncertainties.  
4. Generate a final synthesized summary with citations.  
5. (Optional Phase 2) Visualize contradictions using a simple D3.js or Streamlit dashboard.

## Why MCP?

In a normal RAG system, the LLM only converts retrieved embeddings into natural text — it pulls blindly without verifying sufficiency. In MCP, the LLM becomes an agent that decides when to search, what to read, and how to reason — by using tools exposed through the MCP server.  
  
MCP adds a reasoning layer: the agent checks if context is sufficient, can run multiple steps (search → extract → compare → summarize), and maintains a clear trace of actions and evidence. This makes the system explainable, dynamic, and self-aware — a major leap from static RAG.

## Architecture Overview

User Query  
 ↓  
Claude (LLM) + MCP Client  
 ↓  
[MCP Server] exposes tools:  
 1. search\_papers()  
 2. get\_paper\_details()  
 3. extract\_claims()  
 4. analyze\_contradictions()  
 5. summarize\_findings()  
 ↓  
Agent Workflow:  
 → Search → Retrieve → Extract Claims → Compare → Summarize  
 ↓  
Final Output: Agreements / Contradictions / Uncertainties + Citations

## Tech Stack

LLM: Claude (Pro Desktop + MCP connector)  
Agent Protocol: MCP (Model Context Protocol)  
Backend Framework: FastMCP (Python)  
APIs / Data: OpenAlex API, arXiv API (free and open-access)  
Libraries: requests, json, re, (optional) PyMuPDF for PDF parsing  
Visualization (optional): Streamlit / Plotly / D3.js  
Deployment: FastMCP Cloud (connected to GitHub repo)

## Core MCP Tools

search\_papers(query): Searches OpenAlex for recent papers related to the topic.  
get\_paper\_details(paper\_id): Fetches a paper’s abstract and metadata by ID.  
extract\_claims(papers[]): Uses LLM reasoning to extract the main claims/findings from each abstract.  
analyze\_contradictions(claims[]): Compares claims across papers to find consensus and disagreement.  
summarize\_findings(analysis): Generates a clean, cited final output for the user.

## LLM Prompt Strategy

You are an AI research analyst.  
Your goal is to help users explore and compare research findings.  
You have access to these tools:  
- search\_papers: to find relevant papers.  
- get\_paper\_details: to read abstracts.  
- extract\_claims: to extract each paper’s main findings.  
- analyze\_contradictions: to identify consensus and disagreements.  
- summarize\_findings: to produce a clear final answer.  
  
Workflow:  
1. Search for relevant papers.  
2. Extract 1–3 key claims per paper.  
3. Compare the claims for agreements, contradictions, and uncertainties.  
4. Provide a concise, cited summary.  
  
Always cite papers by title, year, or ID (e.g., OpenAlex:W12345, arXiv:2403.00123).  
Your output must clearly separate:  
- AGREEMENTS  
- CONTRADICTIONS  
- UNCERTAINTIES

## Data Sources

OpenAlex: Scholarly metadata + abstracts (no key needed)  
arXiv: Research preprints (mostly CS/AI)  
Wikipedia (optional): Background info if the user asks for basic context.

## Typical Query Workflow

Example Query:  
“Do recent papers agree on whether small LLMs can outperform large ones through fine-tuning?”  
  
Steps:  
1. search\_papers → finds 5 relevant papers (2023–2025).  
2. get\_paper\_details → retrieves abstracts.  
3. extract\_claims → each paper’s core claim extracted.  
4. analyze\_contradictions → detects patterns.  
5. summarize\_findings → Final answer generated with structure.  
  
Output Example:  
AGREEMENTS:  
- 3 recent papers (2023–2024) find that small fine-tuned models achieve near-parity with large models.  
CONTRADICTIONS:  
- 2 studies report underperformance without additional distillation steps.  
UNCERTAINTIES:  
- The impact of dataset diversity and scaling remains unclear.  
SUMMARY:  
While evidence trends toward parity, performance consistency varies across datasets and methods.  
(Sources: OpenAlex:W34567, arXiv:2404.01234)

## Development Lifecycle

Phase 1 – MVP (Abstract-level): Implement core tools with abstracts.  
Phase 2 – Deep Context: Add section-level reading or PDF parsing.  
Phase 3 – Visualization: Add a front-end to visualize consensus/contradiction graphs.  
Phase 4 – Evaluation: Test on real topics with 10–20 queries.

## Deployment

Build and test locally using FastMCP.  
Connect Claude Desktop using `fastmcp install claude-desktop server.py`.  
Push code to GitHub and deploy via FastMCP Cloud.  
Test with real queries through Claude Desktop (Pro).

## Impact & Use Cases

Students: Quickly understand consensus and controversies in a topic.  
Researchers: Identify gaps or conflicts in their field.  
Institutions: Monitor fast-moving areas and summarize evidence trends.  
General Users: Ask “what’s the scientific consensus on X?” and get sourced, balanced summaries.

## Future Enhancements

Add confidence scoring.  
Show timeline of consensus evolution.  
Integrate citation network graphs.  
Add cross-domain contradiction detection.

## Example Queries

Do recent studies agree on whether LLMs understand syntax?  
What are conflicting findings about AI hallucination mitigation?  
Is there consensus on the environmental impact of training large AI models?  
Have smaller models overtaken large ones on reasoning benchmarks?

## Deliverables Summary

MCP server with 4–5 tools implemented.  
Claude prompt configured for contradiction reasoning.  
Working demo queries with clear “Agreements / Contradictions / Uncertainties”.  
Optional visualization dashboard.  
Deployment on FastMCP Cloud (GitHub-linked).

## Outcome

ResearchMCP transforms a static RAG pipeline into an agentic, self-checking research assistant that understands nuance across multiple studies — helping users move from summaries to scientific understanding.  
  
It’s part researcher, part reviewer — built entirely with open data and free APIs.