Building an Autonomous Research Pipeline

A practical guide to creating a production-ready autonomous business analyst that can research, synthesize, and fact-check information automatically.

Project Architecture Overview

Researcher Layer

Executes search queries and returns 3-5 high-quality candidate URLs using SerpAPI or DuckDuckGo fallback.

Fetcher Layer

Downloads article pages and extracts main textual content using Trafilatura with BeautifulSoup fallback.

Synthesizer

Feeds texts to LLM (OpenAI or local HuggingFace), identifies consistent claims, flags contradictions, and creates citations.

Development Environment Setup

PyCharm Configuration

- Create new project with Python 3.10+ virtualenv
- Install dependencies from requirements.txt
- Set environment variables for API keys

Essential APIs: OPENAI_API_KEY and SERPAPI_API_KEY (optional)

Core Dependencies

Web Scraping

- requests>=2.28
- beautifulsoup4>=4.12
- trafilatura>=1.6.5

AI & ML

- openai>=0.28.0
- transformers>=4.34
- sentence-transformers>=2.2.2

Search & Utils

- serpapi>=2.6.1
- tqdm>=4.65
- torch>=2.1

File Structure & Organization

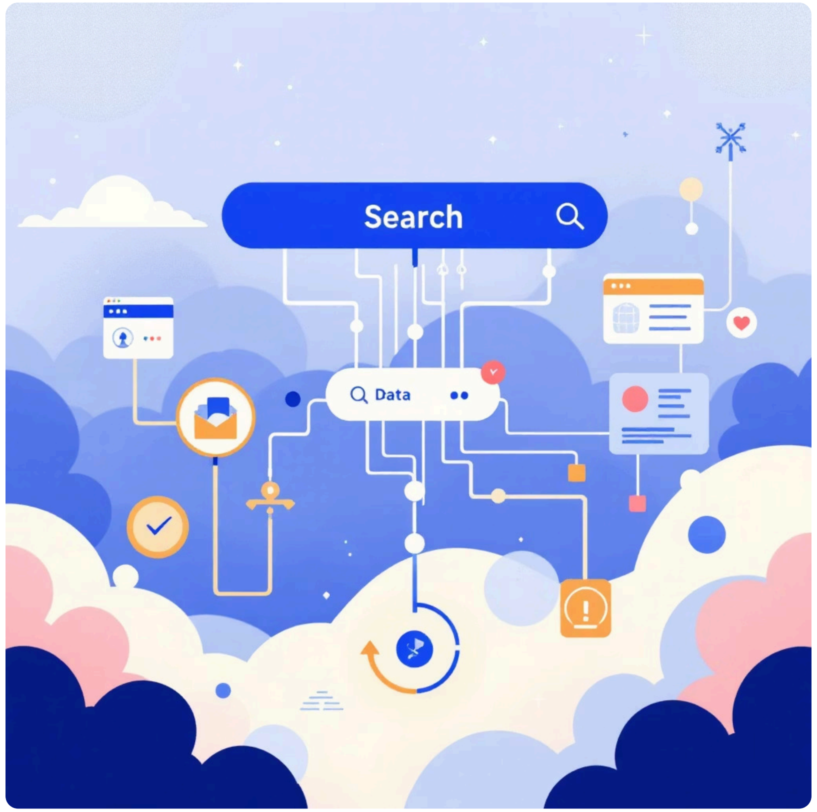
```
autonomous_business_analyst/
├── researcher/
│   ├── searcher.py # search engine interface
│   ├── fetcher.py # fetch & extract text
│   └── utils.py # robots.txt, helpers
├── synthesizer/
│   ├── synthesize.py # LLM orchestration
│   ├── embedding_store.py # similarity helpers
│   └── prompts.py # prompt templates
├── run_all.py # main orchestrator
├── final_answer.txt # generated output
└── requirements.txt
```

Search Strategy Implementation

Multi-Source Search Approach

The pipeline uses multiple search strategies to ensure comprehensive coverage and reliability.

- Primary: SerpAPI for high-quality Google results
- Fallback: DuckDuckGo HTML scraping
- Rate limiting and robots.txt compliance
- Domain filtering for authoritative sources



Fact-Checking & Quality Assurance

01

Redundancy Check

Verify claims appear in 2+ independent sources using semantic similarity matching.

02

Contradiction Detection

Extract definitional sentences and compute pairwise similarity to identify conflicting information.

03

Source Scoring

Assign quality scores based on domain authority, author credentials, and publication date.

LLM Integration Strategy

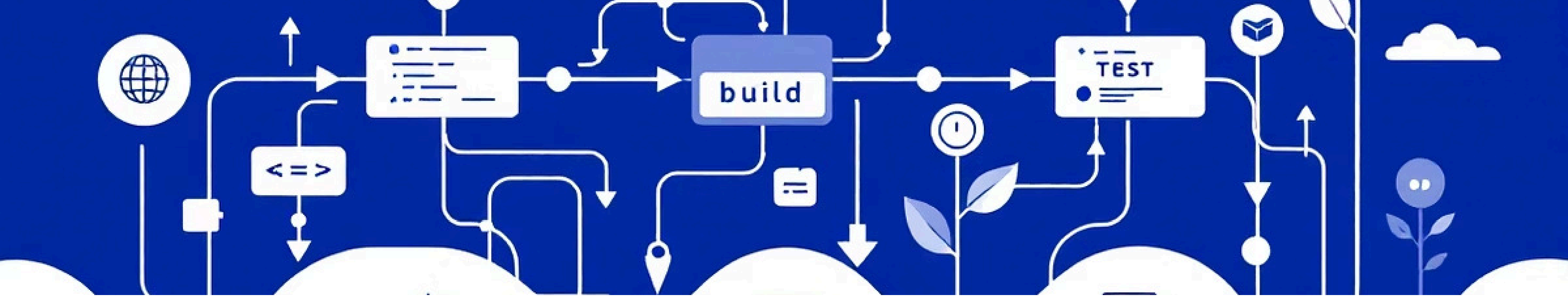
Primary: OpenAI

GPT-3.5-turbo for high-quality synthesis and fact-checking with structured prompts.

Fallback: HuggingFace

Local FLAN-T5 model for offline processing when OpenAI API is unavailable.

Smart fallback system ensures pipeline works even without API access, maintaining autonomous operation.



Testing & Deployment Pipeline

Local Testing

Unit tests for searcher, fetcher, and synthesizer components with known inputs and expected outputs.

GitHub Integration

Automated CI/CD with GitHub Actions to run pipeline and commit results back to repository.

Output Validation

Verify final_answer.txt contains proper citations, reference links, and structured content.

Production Deliverables

Expected Outputs

- Public GitHub repository with complete codebase
- final_answer.txt with inline citations and references
- approach.md documenting design decisions
- README with clear setup and run instructions

Quality Standards

- 300-600 word synthesized answers
- Minimum 3-5 authoritative sources
- Numbered citations matching reference list
- Conservative fact-checking with UNVERIFIED flags