香港城市大學
City University of Hong Kong

Project Report

Data Engineering

- **College**: College of Computing
- **Major and Grade**: Computer Science
- **Team Name**: Null
- **Student Name and Number**:
  - [redacted]
  - [redacted]
  - [redacted]
  - [redacted]
  - [redacted]
  - [redacted]

November 26th 2025

1 Introduction

The automated topic summary page generation platform is a deep-learning based automated news processing system designed to integrate multi-source information on major events. It automatically extracts, analyzes, and integrates key content from news reports, generating structured topic summaries to provide users with a comprehensive overview of events. The platform constructs an end-to-end automated workflow: it crawls news links in parallel using multiple engines, extracts content through intelligent web crawlers, and performs deep analysis using natural language processing technology. The platform also features a deduplication mechanism to ensure the representativeness and completeness of the output information. This platform is suitable for scenarios requiring a rapid grasp of the full picture of complex events, such as media tracking hot topics, researchers gathering industry dynamics, and companies conducting market intelligence analysis. It transforms tedious manual information processing into a highly efficient automated process, significantly improving information processing efficiency. By integrating artificial intelligence technology with engineering practices, the platform effectively addresses the challenge of information overload, providing users with high-quality information services.

2 Technology stack(methodology)

Requests + Beautiful Soup: Enables web crawling and search result parsing across multiple search engines, retrieving news links.

Newspaper3k: A static crawler for quickly extracting news article content, supporting Chinese text parsing.

Selenium: A dynamic crawler for handling JavaScript-rendered web pages, ensuring complete content retrieval.

Fuzzy Wuzzy: Uses fuzzy matching algorithms for title similarity deduplication, avoiding duplicate content analysis.

DeepSeek-V3: Implements AI analysis for text summarization, entity extraction, and information integration using a large language model.

Map-Reduce Architecture: Employs distributed computing to process long texts, splitting before merging to avoid token overruns.

python-dotenv: Manages sensitive configuration information such as API keys and cookies, ensuring security.

HTML Template Engine: Generates structured, visual reports containing core information such as summaries, entities, and timelines.

3 Experiment Workflow

Figure 1 shows an overview of Omnisight's workflow.

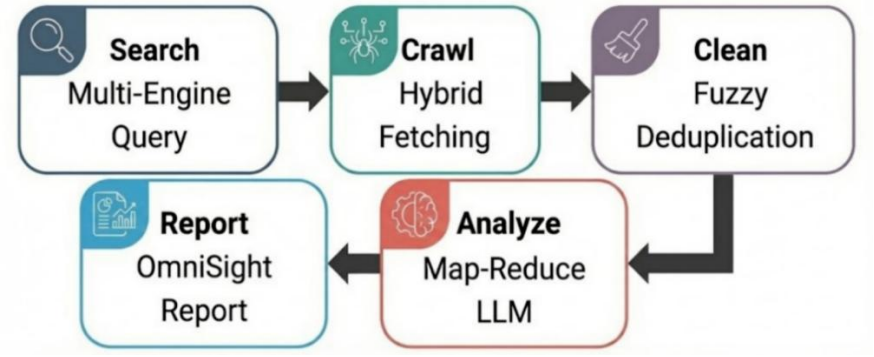


Figure 1: Omnisight's workflow

3.1 News search module

The news search module leverages the collaborative efforts of Google, Baidu, and Bing search engines to ensure comprehensive and reliable information retrieval. The system prioritizes Google search (which requires pre-configured valid authentication information) for the highest quality results. If Google fails to return sufficient results, it automatically switches to Baidu News Search—an engine with unique advantages in Chinese news coverage, requiring no complex verification and exhibiting good stability. When the number of results is still insufficient, Bing search is used as a supplementary source.

During the search process, the system employs a built-in intelligent filtering mechanism to automatically exclude sources with unstable or difficult-to-analyze content and performs real-time URL deduplication to prevent duplicate link collection. The system continuously monitors the number of acquired links and dynamically adjusts subsequent search targets to ensure sufficient results.

Each search engine has an independent fault tolerance mechanism; a single engine failure does not affect the overall process, and the system automatically switches to another available engine to continue operating. Detailed progress feedback is provided throughout the process, allowing users to clearly understand the status at each stage, ensuring operational transparency and facilitating problem diagnosis.

Through this multi-engine, hierarchical search architecture, the system can stably acquire high-quality and diverse news content in various network environments, establishing a reliable data foundation for subsequent in-depth analysis.

3.2 Crawler module

The static crawler module, built on Newspaper3k, efficiently parses web pages and extracts key information such as titles, body text, and publication dates. It supports batch processing of links and generates standardized data. Built-in website blacklists and text quality control mechanisms ensure that only complete professional content is retained.

This module integrates the Fuzzy Wuzzy algorithm, achieving deduplication through fuzzy title matching: automatic filtering when similarity exceeds 85%, supporting the identification of various title variations, and providing detailed deduplication logs. To keep the optimal balance between deduplication effectiveness and information diversity, the actual effects of various similarity thresholds were tested.

Table 1 shows that higher thresholds (>85%) ensure precision but weaken deduplication, 85% achieves the best balance for news aggregation, while 65–85% enforces strict deduplication at the cost of losing diverse perspectives.

Table 1: Test results of various similarity thresholds

Threshold	Effect	Applicable Scenarios
>85%	Almost no false positives, poor deduplication effect, many variants cannot be recognized.	Precise match requirements
85%	Good balance, low false positive rate, effective deduplication, very few related articles may be falsely flagged.	News Aggregator

65%~85%	Thorough deduplication, high false positive rate, important perspectives may be lost.	Strict deduplication of content
<65%	Extremely strict, a large amount of valuable content is filtered.	Special Research

The crawler module has a fault-tolerance mechanism; if static crawling fails, the URL is automatically transferred to the dynamic crawler for processing. The dynamic crawler module, based on Selenium and headless Chrome, simulates real-world access behavior and employs a multi-stage intelligent waiting strategy to balance loading reliability and efficiency. It integrates Beautiful Soup for content parsing, implementing multi-layered text extraction and cleaning, prioritizing paragraph content acquisition, and has a robust resource management mechanism.

The dynamic and static crawlers work closely together, balancing static crawling efficiency with the integrity of dynamic page rendering, providing a comprehensive and reliable data foundation for AI analysis.

3.3 AI analysis

The AI analysis of this project adopts a typical Map Reduce design, which transforms the content of multi-source news into structured information in two stages of "single summary (Map) - global integration (Reduce)", balancing speed and consistency.

3.3.1 Map stage

The goal of the Map phase is to generate concise summaries of approximately 100 words and 3 key points for each article, forming standardized inputs with strong comparability.

In terms of implementation, DeepSeek V3 (model name deepseek chat) is connected through OpenAI SDK to improve output stability at a lower temperature. The prompt clearly requires the output format (abstract+key points), limits the length, and only extracts the first 4000 words of the text to balance performance and relevance. In case of exceptions, the placeholder text "Abstract generation failed..." is returned and a log is recorded to prevent interruption of the process.

Produce multiple 'summary blocks' containing source URLs and lightweight metadata, facilitating subsequent fusion and integration.

3.3.2 Reduce stage

The goal of this stage is to integrate multiple abstracts using a unified JSON structure, generate machine processable conclusions, and support visualization, auditing, and retrieval (such as timelines, entity networks, and topic aggregation).

To ensure that the output is parsed and traceable, strict structural constraints are adopted: JSON is forcibly returned through the SDK's responsive_format, and field names and types are limited. Fields such as main_stummary are predefined in the prompt word, and length and format requirements are clearly defined; At the same time, it is emphasized in the prompt to only use the actual time that is clearly or inferable in the article and attach the source (original URL) to construct the evidence chain; Set the model temperature to a lower value to improve repeatability, and provide a fallback for missing fields when parsing responses (return an empty list or default placeholder) and record inconsistencies or parsing exceptions for manual review.

produce:

Main_Summary: A comprehensive main abstract of 150-200 words, summarizing the core conclusions of multiple reports.

Key_Sub_themes: Various sub themes (such as technology implementation, market response, regulatory situation, industry chain impact), used for theme clustering and tagging.

Key_stities: At least 5 standardized entities (name, company, product, location), supporting relationship diagrams and retrieval.

Timeline: A list of events sorted by time, with each item containing date, event, and source, used for timeline visualization and traceability.

3.4 GUI

Build windows and controls on the GUI front-end using PyQt6, organize the interface through a layout manager, connect user operations and backend threads through a signal slot mechanism, beautify the interface through style sheets, and implement interactive feedback through state management and log areas. Its responsibility is to collect user input, initiate backend tasks, display running logs and results, rather than directly processing data.

The front-end interface is implemented based on the PyQt6 framework. PyQt6 provides cross platform desktop GUI components, where developers inherit QMainWindow to create a main window and place various controls within it. This allows the application to have a standardized window structure, including a title bar, central area, and status management.

In terms of interface layout, the code uses a layout manager to organize controls. QVBoxLayout is used to vertically arrange elements such as titles, input areas, buttons, and log areas; QHBoxLayout is used for horizontal arrangement, such as input boxes, labels, and option combinations. In this way, the interface can remain neat and adaptive, without the need to manually calculate the position of each control. The types of controls are also very diverse. The QLineEdit input box is used to receive event keywords entered by the user; QSpinBox controls the number of articles to be analyzed; QComboBox provides time range selection; The QPushButton button is used to initiate analysis; The log window QTextEdit is used to display information during system operation. These controls are combined together to form a complete interactive interface.

The core mechanism of PyQt is signals and slots. For example, clicking the button will trigger clicked-connect (self. start_analysis), which will call the backend task; The backend thread notifies the frontend to update logs or display results through custom signals log_stignal, success_stignal, and def_stignal. In this way, the front-end and back-end logic are decoupled, with the front-end only responsible for display and interaction.

Interface beautification is achieved through setStyleSheet. The button adopts a gradient background and hover effect, the title is set with font size and color, and the log area uses equal width font, making the overall style more modern. This CSS style sheet makes the PyQt interface no longer default gray controls, but more in line with the aesthetic of modern applications.

Finally, the interface also implements state management. During the analysis process, the button will be disabled and displayed“ ⏳ Analyzing...”, Prevent users from clicking repeatedly; After the task is completed or fails, call reset_ui() to restore the button status. The log area adds real-time information and automatically scrolls to the bottom to ensure that users can see the latest operating status.

3.5 Installation and Usage

3.5.1 Environment Configuration

Clone and enter the project git clone:
https://github.com/SHANECHEN0722/news_seacher.git
cd news_analyzer
Install dependency: pip install - r requirements. txt

3.5.2 Environment variables

Create. env in the project root directory and add OPENAI_API_KEY="sk-your-deepeseek-api-key" (required);

Optional GOOGLE-COOKIE="your Google cookie" to improve Google search quality.

Optional: Get Google Cookie. Open google.com in the browser → F12 → Network → Search for any content → Select the first request → Copy the Cookie field from the Request Header and paste it into. env.

3.5.3 Platform requirements

Python 3.10+; If using dynamic crawlers, Chrome and its corresponding ChromeDriver need to be installed.

3.5.4 Start and Run

Run: Execute ` python main.exe ` in the root directory of the project`.

Usage: Enter keywords on the interface, set the number and time range of captures, and click "Start Analysis". Figure 2 shows the program operation interface.

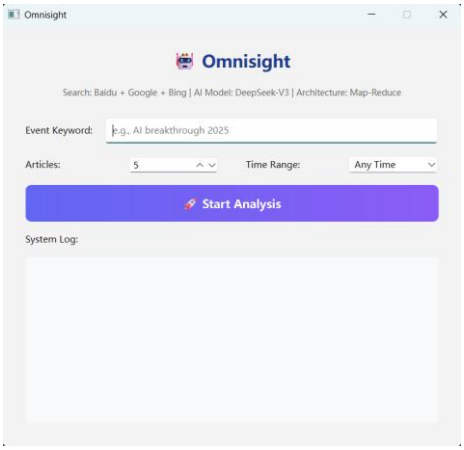


Figure 2: the program operation interface

Report output: The generated HTML report is saved in 'reports/' with a file name format of '{keyword} _ {timestamp}. html'`. Figure 3 shows the HTML interface display of the search results.

Figure 3: the HTML interface display of the search results

After I entered the keyword 'Hong Kong', the above HTML page was generated. This HTML presents the political and cultural status of Hong Kong since its return in 1997 in a structured manner, covering five modules: executive summary, key themes, related entities, event timeline, and reference materials. The content emphasizes the high degree of autonomy and rule of law system under the "One Country, Two Systems" policy, highlighting Hong Kong's competitiveness as a global financial center and the international influence brought by the integration of Chinese and Western cultures, showcasing its unique role in national affairs and the global stage.

4 Summary

This project uses Map Reduce as the skeleton to string multi-source news from "collection de duplication summary structuring visualization" into a closed loop, balancing stability and auditability. The combination of search engines and dynamic crawlers ensures coverage, and the AI integration output with low temperature and multi field constraints ensures consistent and implementable results. GUI provides a smooth operating experience, while HTML reports present key information in dark themes and clear modularity.

For improvement, it is recommended to introduce entity standardization and evidence fragment citation to enhance credibility; Increase conflict detection and thematic hierarchy to enhance analysis depth; Improve export and interaction capabilities, and expand application scenarios in teaching, research, and media monitoring. Overall, the system has the characteristics of being "usable, reproducible, and scalable", making it suitable as a practical baseline for news intelligence aggregation and analysis.

5 Contributions

Xian CHEN(16.6%): Responsible for the overall project architecture design and core module integration, completed system environment configuration and API integration debugging, and wrote complete technical documentation and deployment guidelines.

Han GAO(16.6%): Implemented a three-engine intelligent search system, integrating news retrieval capabilities from Google, Baidu, and Bing, and optimized link deduplication and priority scheduling algorithms.

Yiran YAO(16.6%): Developed a static and dynamic dual-mode crawler system, using Newspaper3k and Selenium to solve the content extraction problem from different websites, and implemented an automatic rollback mechanism.

Wenrui ZHANG(16.6%): Designed a Map-Reduce analysis architecture, implemented article summary generation and information integration based on the DeepSeek model, and optimized prompt word engineering and structured output.

Chunjin ZHU(16.6%): Created a dark-themed visualization reporting system, designed responsive HTML templates, and achieved elegant display of timelines and key information.

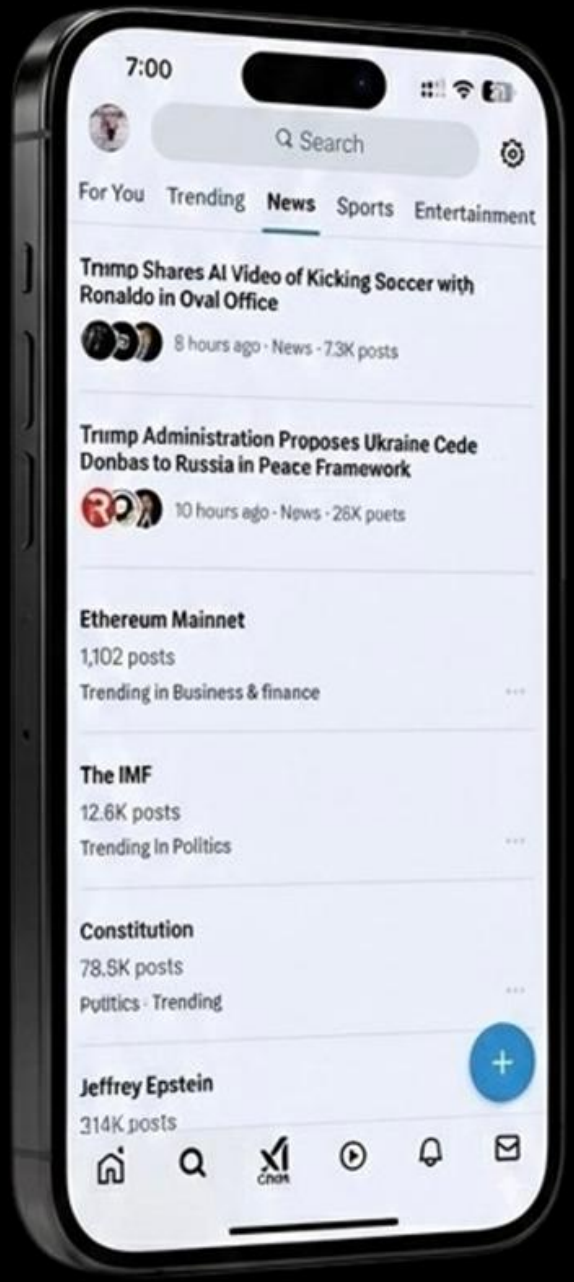
Yi ZHUANG(16.6%): Developed a PyQt6-based user interface, implemented multi-threaded task processing and real-time progress feedback, and provided a smooth interactive experience.

OmniSight

From Chaos to Clarity.

TOPIC 2: AUTOMATED TOPIC SUMMARY PAGE GENERATION

Team: Null | Reporter: Han Gao



07:00 AM. Chaos.

The Information Overload

Imagine waking up to a breaking event. You check Twitter: 500 new posts. You check Google: 20 articles with conflicting headlines. The data is there, but the clarity is missing.

In the age of big data, our biggest problem isn't finding information—it's synthesizing it instantly.

Why OmniSight?

For Decision Makers

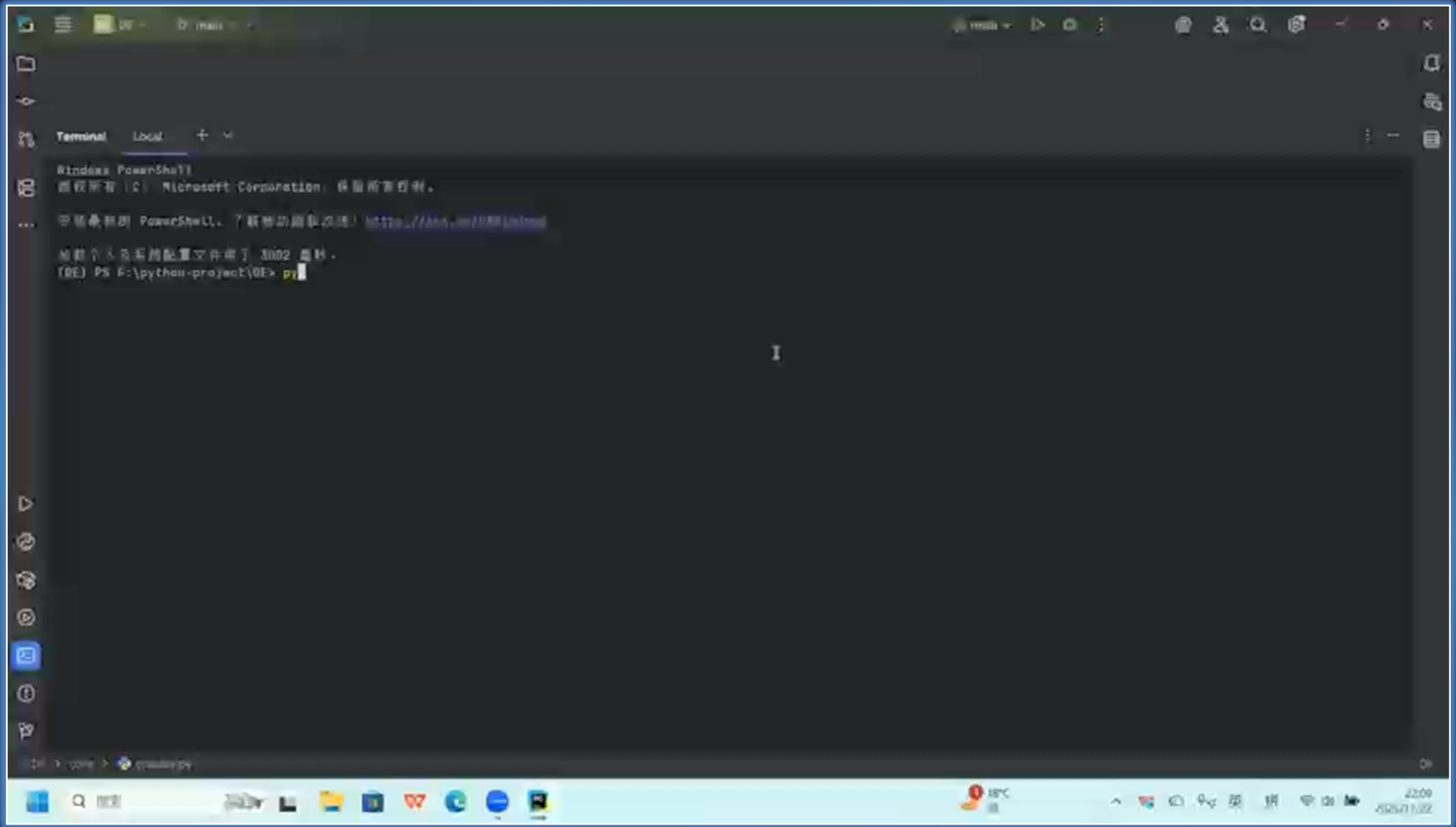
Researchers, analysts, and users who need to grasp complex situations in seconds, not hours.

The Friction

Manual compilation is slow. News fragmentation across platforms makes it impossible to see the full picture.

"Three search engines.
One intelligent brain.
Infinite clarity."

The OmniSight Vision


Terminal
Local
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。
(DE) PS F:\python-project\DE> py

System Architecture

1. Intelligent Search

PYTHON + REQUESTS

Multi-Source Ingestion

OmniSight doesn't rely on a single source.

Benefit: Elimination of source bias and maximum event coverage.

2. Hybrid Crawling

Static Mode

Uses **Newspaper3k**. Ultra-fast. Handles most of standard news sites. Low resource consumption.

Dynamic Mode

Uses **Selenium**. Automatically activates when static crawling fails. Renders JS-heavy sites to capture elusive data.

3. Smart Deduplication

85%

Similarity Threshold

FUZZYWUZZY LIB

Signal to Noise Ratio

News outlets often copy-paste content. We use the Levenshtein Distance algorithm to compare article titles.

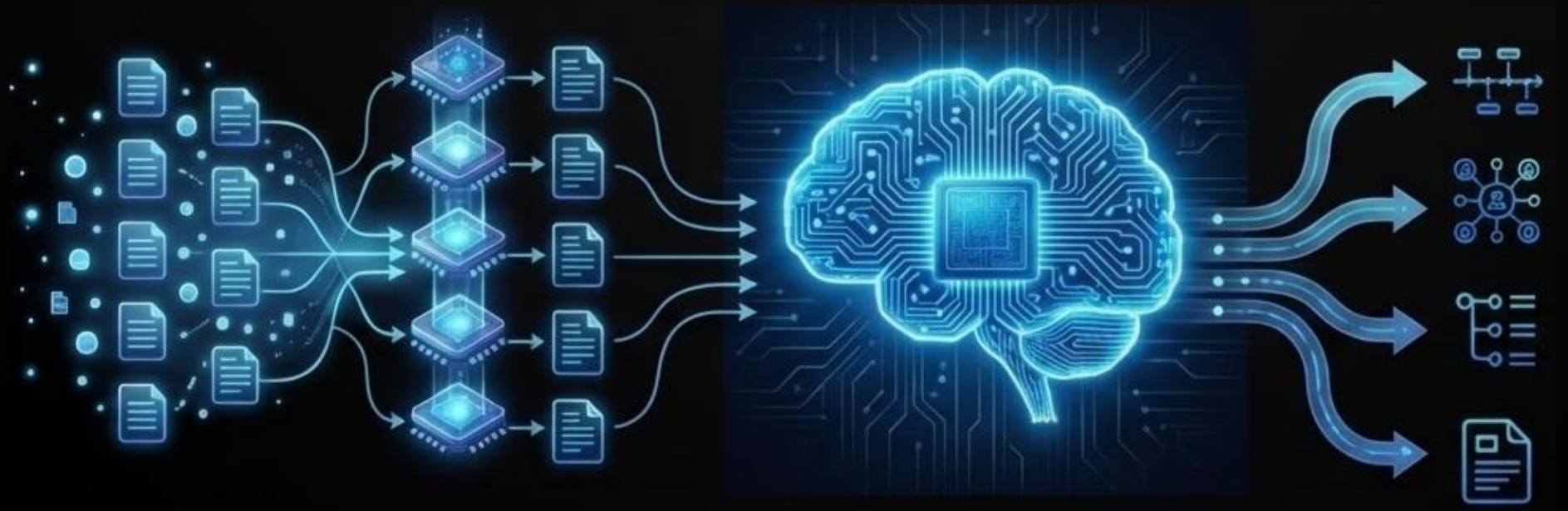
If Article A is >85% similar to Article B, OmniSight discards the duplicate, ensuring the LLM only reads unique, high-value content.

4. Map-Reduce Analysis

Map Phase

Reduce Phase

Output



Summarize

Aggregate

Future Horizons

Knowledge Graphs

Moving beyond text summaries to visual node-link diagrams (Neo4j) to show relationships between entities.

Real-Time Streaming

Replacing batch processing with WebSocket-based real-time feeds for instant breaking news updates.

responsibilities and Contributions

Name	contributions	responsibilities
Xian CHEN	16.6%	Overall project architecture design and core module integration
Han GAO	16.6%	Implemented the three-engine intelligent search system
Yiran YAO	16.6%	Developed a dual-mode (static and dynamic) web crawler system
Wenrui ZHANG	16.6%	Designed the MapReduce analysis architecture
Chunjin ZHU	16.6%	Created the visualization reporting system
Yi ZHUANG	16.6%	Developed the user interface

Q & A

Thank you.

github.com/SHANECHEN0722/news_seacher