Web3 Daily Newsletter Bot

An **automated multi-agent system** that scrapes news from top Web3 sources, summarizes them using LLM agents, and sends a **daily digest to Telegram**.

Objective

Automate the process of gathering, deduplicating, summarizing, and broadcasting the **top 10 Web3 articles daily**.

Project Structure

```
web3newsletter/
├── bots/
│   └── telegram.py            # Telegram bot integration
│
├── newsletter/
│   ├── composer.py            # Formats the newsletter content
│   └── templates.py           # Defines reusable formatting blocks
│
├── processors/
│   ├── deduplicator.py        # Removes similar/duplicate articles
│   └── summarizer.py          # Summarizes content using LangChain
│
├── scrapers/
│   ├── __init__.py
│   ├── bankless.py
│   ├── coindesk.py
│   ├── cointelegraph.py
│   ├── decrypt.py
│   ├── theblock.py
│   └── base_scraper.py        # Abstract class for scraper templates
│
├── scheduler.py               # Orchestrates the end-to-end pipeline
├── test_one_source.py         # For testing individual scrapers
├── config.py                  # Stores Telegram token and constants
├── README.md                  # Setup and usage guide
├── SUMMARY.md                 # Report of design decisions and issues
└── requirements.txt           # Python dependencies
```

Features

Scrapes from top sources:

- CoinDesk
- CoinTelegraph
- Decrypt
- The Block
- Bankless

Deduplicates articles using **cosine similarity** on titles
Summarizes articles using **LangChain LLM agent**
Formats newsletter using a markdown-safe layout
Sends newsletter to a **Telegram group**
Supports **simulated 2-day digest** for testing

Tech Stack

- Python 3.8+
- LangChain (OpenAI API)
- BeautifulSoup + Requests
- Pydantic
- Telegram Bot API
- (Optional) VectorDB for semantic clustering

Installation

git clone https://github.com/yourusername/web3newsletter.git
cd web3newsletter
python -m venv venv
venv\Scripts\activate
pip install -r requirements.txt

Edit config.py:

TELEGRAM_TOKEN = "your_bot_token"
TELEGRAM_CHAT_ID = "your_group_id"

Running the Bot

🔹 **To run for one day (all sources):**

python scheduler.py

🔹 **To simulate two days in a row:**

python scheduler.py
(This just runs the pipeline twice; actual historic fetching is not implemented.)

🔹 **To test one source only:**

python test_one_source.py

Telegram Demo

Join test group:
👉 Telegram Group Invite

Challenges Faced

1. **Telegram Markdown Parsing (400 errors)**
   - Telegram's MarkdownV2 is overly strict. Escaping characters like _, -, (,) was error-prone.
   - *Fix:* Switched to plain text + emojis for bullet points.
2. **Telegram Message Limit (4096 chars)**
   - Full summaries of 10 articles exceeded limits.
   - *Fix:* Added message chunking with smart splitting.
3. **LangChain Latency**
   - LangChain summarization was slow (30–60 sec/batch).
   - *Fix:* Reduced prompt length and limited summarization to top 10 articles.
4. **Duplicate News Across Sources**
   - Multiple articles with same title or content.
   - *Fix:* Cosine similarity used on normalized titles for deduplication.

Architectural Highlights

- **Modular by Responsibility**

    - scrapers/: Each source's crawler
    - processors/: Deduplication and summarization
    - newsletter/: Formatting logic
    - bots/: Telegram integration
- **Stateless and Lightweight**
    - No database used. Each run starts fresh.
    - Easy to deploy on serverless platforms
- **Two-Day Simulation Support**
    - Same-day content simulated as two separate days (for testing)
- **LangChain + OpenAI**
    - Used for high-quality summarization via agentic prompts

Scope for Improvement

Area	Idea
Persistent Storage	Store articles to deduplicate across days
Semantic Deduplication	Use VectorDB (FAISS, Chroma) for better clustering
Smart Message Splitting	Dynamically break up content into chunks without mid-article breaks
Date Filtering	Scrape based on real article publish timestamps
Admin UI/Dashboard	Preview/edit newsletter before pushing (Streamlit or Flask)
Archival Support	Save sent newsletters for history and re-use
Improved Prompting	Add prompt engineering and few-shot examples for better summaries

Conclusion

Despite external API constraints (Telegram + OpenAI), this system successfully:

- Scrapes 5 reliable sources
- Deduplicates with semantic logic
- Summarizes via LangChain agents
- Sends updates via Telegram
- Supports simulated 2-day testing

This codebase is **modular, well-structured, and production-ready**, offering a powerful foundation for building a Web3-focused daily news digest automation.