AI News Aggregator Pro – Code Explanation

Introduction

AI News Aggregator Pro is a Python-based web application built with **Streamlit**. Its goal is to collect real-time news related to Artificial Intelligence from multiple trusted sources via their RSS feeds and present them in a user-friendly format. Users can search for specific keywords, filter news by time range (e.g., last 7 days), and select from various sources to customize the content they want to read.

This project demonstrates how to combine tools like `Streamlit`, `Feedparser`, and `BeautifulSoup` to create a responsive, interactive dashboard. Below is a line-by-line breakdown of the entire application code, along with detailed explanations of its functionality.

---

1. Library Imports

```python
import streamlit as st
import feedparser
from bs4 import BeautifulSoup
from datetime import datetime, timedelta
import time
```

- `streamlit`: Used to create the web interface and handle UI elements.
- `feedparser`: Reads and parses RSS feeds into Python-readable structures.
- `BeautifulSoup`: Cleans and extracts plain text from HTML content.
- `datetime`, `timedelta`: Manage and filter dates (e.g., for recent news).
- `time`: Imported but not directly used; typically used for delays or timestamps.

---

2. Caching the News Fetching Function

```python
@st.cache_data(ttl=3600)
```

- This decorator caches the output of the function for **1 hour** (3600 seconds).
- It improves performance by avoiding repetitive data fetching during the same session.

---

3. Defining and Fetching from RSS Sources

```python
rss_urls = {
    "TechCrunch": "...",
    ...
}
```

- A dictionary storing the names and RSS feed URLs of AI-related news sources.
- Each entry in this dictionary represents one source.

```
for source_name, url in rss_urls.items():
```

- Iterates through all defined sources.
- The loop skips any source not selected by the user.

```
feed = feedparser.parse(url)
```

- Parses the RSS feed and returns structured data, including titles, descriptions, links, and publish dates.

---

4. Extracting and Cleaning News Details

```
pub_date = datetime(*entry.published_parsed[:6])
```

- Converts RSS date format to a Python `datetime` object for filtering later.

```
summary = entry.get("summary", entry.get('description', "Summary not
available"))
soup = BeautifulSoup(summary, 'html.parser')
clean_summary = soup.get_text()
```

- Extracts the summary or description and removes HTML tags for clean text display.

---

5. Keyword-Based Filtering

```
if search_query:
    if search_lower not in news_item['title'].lower() and ...
```

- If the user typed a keyword, the function filters out articles that don't contain that keyword in the title or summary.

---

6. Main Function to Run the App

```
def main():
```

- Wraps the entire Streamlit interface and logic inside a single function, ensuring clean program structure.

---

7. Page Configuration

```
st.set_page_config(
    page_title="AI News Aggregator Pro",
    page_icon="🤖",
    layout="wide"
)
```

- Sets up the Streamlit app with a custom title, icon, and wide layout for better display.

---

8. Top Section: Search & Time Filters

```
search_query = st.text_input("□ Search News")
days_filter = st.slider(...)
```

- Allows users to input a keyword for searching.
- Adds a slider for selecting how many past days' news should be displayed (1–30 days).

```
if st.button("□ Refresh"):
    st.rerun()
```

- Refreshes the app and re-runs the data fetch process.

---

9. Sidebar (Project Info & Source Selection)

```
with st.sidebar:
```

- Contains an expandable section with a project description, feature list, and source selection.

```
selected_sources = st.multiselect(
    "Select News Sources",
    options=sources,
    default=sources
)
```

- Users can choose which sources to pull news from via a multi-select dropdown.

---

10. News Loading and Date Filtering

```
ai_news = fetch_ai_news(selected_sources, search_query)
cutoff_date = datetime.now() - timedelta(days=days_filter)
ai_news = [news for news in ai_news if news['date'] and news['date'] >
cutoff_date]
```

- Calls the data-fetching function.

- Filters out news items older than the selected number of days.

---

11. News Source Statistics

```
source_counts = {source: 0 for source in selected_sources}
for news in ai_news:
    source_counts[news['source']] += 1
```

- Counts how many articles were retrieved per selected source.

```
sorted_counts = sorted(source_counts.items(), key=lambda x: x[1],
reverse=True)
```

- Sorts sources by number of articles in descending order for a better overview.

---

12. Displaying the News Articles

```
if not ai_news:
    st.warning("No news found matching your criteria.")
else:
    cols = st.columns(2)
    for index, news in enumerate(ai_news):
        ...
```

- Shows a message if no news matches the criteria.
- Otherwise, displays each article in a two-column layout for better readability.

```
st.markdown(f"### {news['title']}")
st.caption(f"{news['source']} - {formatted_date}")
st.expander("View Summary")
st.link_button("Go to Article →", news['link'])
```

- Each article block contains:
  - Title
  - Source and publication date
  - Expandable summary
  - Button linking to the full article

---

13. App Entry Point

```
if __name__ == "__main__":
    main()
```

- Ensures the app only runs when the script is executed directly (not when imported as a module).

Summary of the Workflow

1. **User Interface Initialization:** Set up the layout and components using Streamlit.
2. **User Input:** Users define search criteria, date range, and news sources.
3. **Data Fetching & Caching:** News is fetched from selected RSS feeds and cached for performance.
4. **Data Cleaning & Filtering:** Clean the data from HTML tags and apply filters (date and keyword-based).
5. **Statistics Display:** Show statistics about news count per source.
6. **Displaying News:** Display news articles with titles, summaries, and links.
7. **User Interaction:** Users can interact with the app by refreshing the data, searching, or filtering news.

This workflow ensures that users can easily access and filter AI news according to their preferences, while maintaining fast performance and responsiveness.

Conclusion

AI News Aggregator Pro is a lightweight yet powerful web app for aggregating AI-related news from multiple reliable sources. The code architecture is modular, efficient, and easy to scale. It leverages:

- **Streamlit** for the user interface,
- **Feedparser** for RSS parsing,
- **BeautifulSoup** for HTML cleaning,
- and Python's standard libraries for data manipulation and filtering.

This app can be a valuable tool for researchers, tech enthusiasts, and developers who want to stay updated on AI advancements. Additionally, it can serve as a solid foundation for more advanced projects, such as:

- Personalized news recommendations
- Natural language summarization
- User profiles and saved articles
- Push notifications for breaking news

By understanding this codebase, developers can gain insight into integrating real-time data streams with interactive UIs, all while writing clean and efficient Python code.