ChatGPT

자동 뉴스 모니터링 시스템 개발 계획 분석

주요 요구사항 요약

- **키워드 입력→뉴스 수집:** 사용자가 주제(예: 증시)를 입력하면 Google News RSS 및 네이버 뉴스 검색 API 등을 통해 관련 기사를 비동기로 수집.
- **주기적·수동 실행:** 하루 2회(자정, 정오)에 자동 실행하고 필요 시 수동 트리거 가능하도록 스케줄링.
- **로컬 LLM 분석:** 수집된 기사 텍스트를 한글 대응 LLM(예: Llama-2-Ko, KoGPT 등)에 입력하여 핵심 주제·연관성 분석. 가능한 외부 API 호출을 최소화하고, 로컬 GPU 환경(예: 80W GTX 4070)에서 동작.
- **시각화/요약 출력:** NotebookLM 같은 인터랙티브 그래프 형태로 주제 간 연관성을 시각화하거나, 이메일 요약 알림 형태로 결과 제공.
- **개발 기간:** 2025년 6월 15일~6월 30일 (2주)로, 구체 일정과 단계별 구현 계획이 포함되어야 함.

각 요구사항을 구현하기 위해 고려해야 할 요소, 기술 난이도, 병목 구간, 적절한 도구 등을 아래에 상세히 분석한다.

1. 뉴스 수집 전략

- **구글 뉴스 수집:** Google News는 공식 API가 없으므로 RSS 피드를 활용하거나 서드파티 라이브러리를 이용한다. 예를 들어 Python 패키지 **Google-News-Feed**는 Google 뉴스 RSS를 `aiohttp`, `lxml` 기반으로 비동기 처리해준다 [1]. 또는 **GNews** 라이브러리는 Google News RSS를 파싱해 JSON 형태로 결과를 제공하며, `newspaper3k` 라이브러리와 연동해 기사 전문도 쉽게 가져올 수 있다 [2] [3]. 이들 패키지를 사용하면 비동기로 여러 쿼리를 동시에 요청할 수 있어 효율적이다.
- **네이버 뉴스 수집:** 네이버는 뉴스 검색용 공개 API를 제공한다. 키워드를 쿼리하면 뉴스 목록(제목, URL 등)을 JSON/XML로 반환한다 [4]. 일일 호출 한도(25,000회)를 주의하되, 일반적인 사용량(하루 수십~수백회)에는 무방하다. 네이버 API 사용 시 앱 등록 후 발급받은 Client ID/Secret을 HTTP 헤더에 포함해야 한다 [4].
- **스크래핑 vs API:** 단순 HTML 크롤링은 태그 구조 변경과 IP 차단 위험이 크므로, 가능한 공식 API를 사용한다는 점에 유의해야 한다. 실제 구현 사례에서도 “자동화된 프로그램이 잦은 횟수로 HTML을 긁어가면 IP를 차단당할 수 있으므로 정식 API를 사용”하는 방식으로 구현되었다 [5].
- **비동기 처리:** Python의 `asyncio` / `aiohttp` 등을 이용하면 다수의 뉴스 요청을 병렬 처리할 수 있다 [1]. 예를 들어 구글 뉴스 RSS 라이브러리에서 `resolve_internal_links=True` 옵션을 사용하면 내부 링크 해제를 비동기 방식으로 처리한다 [6]. 네이버 API 호출도 `aiohttp`로 감쌀 수 있으며, 구글과 네이버 요청을 동시에 처리하도록 구성할 수 있다.

2. 스케줄링 및 실행 환경

- **스케줄링:** 매일 00시와 12시에 자동으로 실행되도록 `cron`이나 Python의 **APScheduler** 등을 사용할 수 있다. APScheduler를 쓰면 크론 스타일(`hour=0, minute=0` 등)로 작업을 예약할 수 있고, 수동 실행도 커맨드 입력 한 줄로 가능하다. 트리거 실패 대비 로그 관리와 오류 알림 체계를 구축하면 안정성을 높일 수 있다.
- **환경 준비:** 로컬 GPU 환경(80W GTX 4070)에서 LLM이 돌아가야 하므로, CUDA/cuDNN 설치 및 PyTorch/Transformers 등의 라이브러리를 세팅한다. LLM 모델은 사전 다운로드하여 사용하며, 필요한 경우 4비트·8비트 양자화(예: BitsAndBytes)를 활용해 VRAM 사용량을 줄인다 [7]. 코루틴 작업과 LLM 추론을 동시에 돌리면 GPU 메모리가 급증할 수 있으므로, 메모리 관리에 유의하며 오프로드를 검토한다.

3. 데이터 처리 및 저장

- **텍스트 추출:** 검색 API나 RSS로는 기사 URL과 요약만 얻으므로, 실제 기사 전문을 보려면 링크된 원문을 크롤링해야 한다. 이를 위해 **newspaper3k**나 **BeautifulSoup** 등을 사용해 HTML에서 텍스트만 추출한다. 예를 들어 GNews 라이브러리의 `get_full_article()` 메서드는 내부적으로 newspaper3k로 `Article` 객체를 생성하여 `text` 필드에 전문을 담아준다 [3]. 텍스트 전처리(불필요 공백/HTML 태그 제거 등)를 거쳐 LLM 입력에 적합한 형태로 저장한다.
- **저장 포맷:** 수집 기사 데이터(제목, URL, 날짜, 전문)와 LLM 분석 결과는 데이터베이스(MySQL, SQLite 등), 혹은 JSON/CSV 파일에 저장할 수 있다. 추가로 이후 주제 연관 분석을 위해 텍스트 임베딩을 미리 계산하여 벡터 DB(예: Faiss)나 pandas DataFrame에 저장해 두는 것도 고려할 수 있다.

4. LLM 분석 (국내 한국어 LLM)

- **모델 선택:** 한글 처리를 위해서는 KoGPT-2(약 4.8B), Llama-2-Ko(예: 7B) 등 한국어 혹은 한국어 코퍼스를 포함한 LLM이 적합하다. 특히 7B 규모 Llama-2-Ko 모델은 한국어 말뭉치 추가 학습을 통해 한국어 이해력이 개선된 예이며 [8], 12GB급 GPU에서도 양자화하여 운용 가능하다. 실측 RTX 4070(12GB VRAM)에서는 양자화된 13B 모델도 구동 가능하므로 [7], 개발 중에는 Llama-2-Ko-7B나 코사전 훈련된 KoGPT-2 4.8B 정도를 우선 고려한다.
- **추론 환경:** Transformers 라이브러리를 이용해 로컬 추론을 구현하고, 입출력 인터페이스(Python 함수)를 만든다. GPU 메모리 압박을 줄이기 위해 4비트 양자화(예: LLama.cpp의 GGUF 또는 Huggingface의 BitsAndBytes)나 문맥 길이 조정 등을 검토해야 한다 [7]. 저전력 노트북 환경이므로 너무 큰 모델보다 7B~13B 범위 모델이 타당하다.
- **분석 방식:** 수집된 기사 텍스트를 한꺼번에 요약하도록 LLM에 묻거나, 기사의 핵심 키워드/주제를 추출하도록 한다. 예를 들어 "이 기사에서 중요한 키워드를 추출하라"고 프롬프트하거나, 문서 유사도 분석을 위해 임베딩을 계산할 수 있다. 단, GPU 연산 속도가 느릴 수 있으므로, 필수 수준으로 요청을 최소화하여 처리한다.

5. 주제 연관성 탐색 및 시각화

- **연관 주제 탐색:** LLM을 활용해 기사 간 연관도나 공통 개념을 추출하는 것은 고차원 작업이다. 단순한 방법으로는 각 기사의 키워드를 비교해 연결망을 만들거나, 문서 임베딩 유사도(코사인 유사도)를 계산해 클러스터링할 수 있다. 예를 들어 **Sentence Transformers** 등을 써서 문장 임베딩을 구하고, `scikit-learn`의 유사도 측정으로 관련 기사끼리 묶는 방식이 있다. 이 과정은 주제어 간 연결망(knowledge graph) 구축의 전처리가 될 수 있다.
- **시각화:** 분석 결과를 그래프로 보여주기 위해 **NetworkX**로 노드(뉴스, 키워드)와 엣지를 정의하고, **Pyvis** 같은 라이브러리로 인터랙티브 네트워크 그래프를 만든다. Pyvis는 Streamlit 등과 쉽게 연동되어 웹 브라우저에서 회전·확대 가능한 네트워크를 지원한다 [9]. 예를 들어 Streamlit 앱에서 Pyvis를 사용하면 간단한 코드만으로 데모 그래프를 보여줄 수 있다. NotebookLM처럼 매우 정교한 UI 구현은 어려우므로, 우선 Pyvis/Streamlit 형태나 대체로 Graphviz 이미지 삽입 정도로 시각화 기능을 시작할 수 있다.
- **이메일 요약:** 그래프 인터페이스 구현이 어렵다면, LLM으로 핵심 내용을 요약한 뒤 Python `smtplib` 등을 이용해 이메일로 정해진 수신자에게 보고서를 발송할 수 있다. 요약문에는 주요 키워드, 새로운 소식, 다음 관심 키워드 제안 등을 포함할 수 있다. 자동 메일은 CRON이나 스케줄러와 연결해 매일 적시에 보내도록 한다.

6. 기술적 난이도 및 병목 구간

- **크롤링 안정성:** Google News는 비공식 루트를 써야 하므로 캡차 혹은 IP 차단 위험이 있다. 네이버는 공식 API를 써야 하지만, API 응답이 하루 한도에 걸리거나 네트워크 불안정 시 장애가 발생할 수 있다. 특히 많은 양의 요청을 비동기로 보내면, 예기치 않은 HTTP 오류도 고려해야 한다.
- **LLM 성능:** 12GB GPU에서 LLM 추론은 메모리와 연산 속도가 병목일 수 있다. 모델 크기가 커질수록 토큰 처리 속도가 급격히 느려지며, 한국어 모델은 해당 모델만의 특성으로 추가 메모리(토크나이저 등)가 필요하다. 양자

화 등 최적화가 필수이며, 추론 중 OOM(메모리 부족) 이슈에 대비해 최소화된 입력 텍스트와 배치 사이즈를 사용해야 한다 [7].

- **시간 제약:** 2주라는 짧은 일정에 개발·테스트·통합까지 완료해야 하므로, 기능을 모두 완성하기는 매우 빡빡하다. 특히 시각화·인터랙티브 UI 개발은 상당한 시간이 소요될 수 있다. LLM 조정, 에러 처리, 예외 상황 점검 등도 충분한 시간이 필요하므로 일정 관리가 핵심이다.
- **병목 처리:** (1) LLM 추론 지연(각 문서 요약에 수 초~수십 초 소요), (2) 비동기 작업 동기화와 에러 핸들링, (3) 그래프 노드 수가 늘어날 경우 시각화 복잡도 등이 주요 병목이다. 특히 노트북 환경이므로 멀티스레딩/멀티프로세싱 대신 단일 프로세스로 구현할 경우 동시성 문제도 고려해야 한다.

7. 일정 제안 및 단계별 실행 방안

현재 제시된 **2025년 6월 15일~30일(2주)** 일정은 매우 촉박하므로, 기능을 우선순위별로 나누어 점진적으로 구현하는 것이 바람직하다. 아래는 예시 일정이다. 필요에 따라 실제 팀 규모와 경험에 맞춰 조정하되, 핵심 기능(뉴스 수집·LLM 분석·기본 결과 요약) 우선 구현을 권장한다.

기간	주요 작업	비고 (위험도 등)
6/15~6/16	개발환경 구축, 요구사항 상세화	Python 환경·라이브러리 설치, Naver API 키 발급
6/17~6/19	뉴스 수집 모듈 구현 (구글/네이버)	Google News RSS 라이브러리 및 Naver API 활용 검사 [1] [4]. 동시 요청 테스트 (에러 처리 포함).
6/20~6/21	기사 본문 파싱 및 저장	`newspaper3k` 나 `BeautifulSoup` 로 본문 추출. 결과 DB/파일 저장. (예시처럼 CSV→HTML→텍스트 처리 [10])
6/22~6/23	LLM 분석 모듈 개발	Llama-2-Ko 등 모델 다운로드·로드. 기사 텍스트 요약/키워드 추출 테스트. VRAM 최적화 검증 [8] [7].
6/24~6/25	스케줄링·자동화 기능 구현	APScheduler/Cron 설정, 수동/자동 실행 모드 구현. 로그 기록 체계 마련.
6/26~6/28	결과 시각화 및 요약 출력	Pyvis/Streamlit으로 간단 그래프 데모 제작(선택) [9]. 이메일 요약 기능(뉴스 요약, 상위 키워드 포함) 구현.
6/29	통합 테스트 및 버그 수정	전체 파이프라인 흐름 점검. 예외 상황(네트워크 장애, LLM 오류 등) 처리.
6/30	완성도 점검 및 발표 자료 준비	결과 검토, 문서화. 추가 개선 포인트 정리.

위 일정은 **MVP(핵심 기능)** 중심으로 구성하였다. 특히 시각화(그래프 UI)는 초기 버전에서 제외하고 이메일 요약으로 대체할 수 있다면 일정 부담을 줄일 수 있다. 예를 들어 **1단계**에서 뉴스 수집과 텍스트 추출, **2단계**에서 LLM 분석 결과 확인, **3단계**에서 기본적인 보고서 출력(이메일/정적 그래프)까지 구현한다. NotebookLM 스타일의 복잡한 인터페이스는 이후 **유지보수 단계**로 미루고, 먼저 핵심 기획을 작동시키는 것을 목표로 한다.

결론

제안된 자동 뉴스 모니터링 시스템은 기술적 요소가 많아 2주 내 완전 구현은 도전적이다. 특히 로컬 LLM 연동과 인터랙티브 시각화는 별도 기술 검증이 필요한 부분이다. 따라서, 일정이 촉박하다면 **단계별 개발**(뉴스 수집 → LLM 분석 → 결과 요약/알림)로 기능을 나누고, 일단 동작하는 파이프라인을 구축한 뒤 점차 고도화할 것을 권장한다. 병목이 예상되는 부분(스크래핑/IP 차단, LLM 메모리 한계 등)에는 사전 대처(공식 API 활용, 모델 양자화 등)가 필요하다. 위 예시 일정과 분석을 참고하여, 현실적인 범위 내에서 기능 우선순위를 조정하는 것이 성공적인 개발의 관건이다.

참고: 구글 뉴스 RSS 처리 예시 [1] [2], 네이버 뉴스 검색 API [4], 크롤링 대 API 논의 [5], 현업 구현 사례 [10] [11], 한국어 LLM 모델 정보 [8], GPU 사양별 LLM 운용 가이드 [7], Pyvis 시각화 사용 예 [9] 등에서 관련 정보를 참고하였다.

---

[1] [6] GitHub - LLukas22/Google-News-Feed: A simple python library to consume the google news rss feed
https://github.com/LLukas22/Google-News-Feed

[2] [3] gnews · PyPI
https://pypi.org/project/gnews/

[4] 검색 > 뉴스 - Search API
https://developers.naver.com/docs/serviceapi/search/news/news.md

[5] [10] [11] LLM을 사용한 뉴스 검색 후 분석 프로그램
https://www.gpters.org/chatbot/post/puw7Ir5GuC5b34u

[7] Best Local LLMs for Every NVIDIA RTX 40 Series GPU
https://apxml.com/posts/best-local-llm-rtx-40-gpu

[8] beomi/llama-2-ko-7b · Hugging Face
https://huggingface.co/beomi/llama-2-ko-7b

[9] Interactive Networks (graphs) with Pyvis - Show the Community! - Streamlit
https://discuss.streamlit.io/t/interactive-networks-graphs-with-pyvis/8344