아래는 “편집장이 하루 한 키워드” 역사 큐레이션형 퀄리뉴스를 구축하기 위한 구체적 개발 가이드입니다. 모듈 설계, 작업 단계, 일정, 역할 분담, 산출물 등을 체계적으로 정리했습니다.

---

1. 설계 및 준비 단계

1.1 요구사항 정리

- **키워드 기반 뉴스**: 하루 한 키워드를 중심으로 공식 뉴스, 학술 논문, 표준/정책 문서, 커뮤니티 글 등 다양한 자료를 모아 히스토리형 뉴스 콘텐츠를 발행하는 것이 목표입니다.
- **기간·자료 범위**: 과거 10년 이상을 포괄하는 자료까지 포함하며, 키워드별로 데이터베이스를 축적합니다.
- **품질 관리**: 유형별로 적합한 품질 평가를 수행해 신뢰도 낮은 자료를 제외합니다.
- **편집자 인터페이스**: 자료 검토·승인·발행을 위한 UI가 필요합니다.

1.2 기술 아키텍처

- Python 기반 크롤러 및 백엔드
- 데이터 저장: JSON → 초기 SQLite 또는 PostgreSQL → 검색 최적화 필요 시 Elasticsearch
- 작업 스케줄링: cron, APScheduler
- 간단한 관리자 UI: Flask/FastAPI + Vue/React (선택)

---

2. 모듈 설계

아래 모듈을 독립적으로 구현하고, 단계별로 통합합니다.

2.1 source_loader.py

- feeds/official_sources.json, feeds/community_sources.json, feeds/paper_sources.json 등 소스 파일을 로드하여 도메인, 유형, API URL 등을 제공.
- 소스 파일 형식 예:
- ```
  {
    "official": {
      "ipc": "https://ipc.org/rss",
      "iconnect007": "https://iconnect007.com/smt/rss"
    },
    "community": {
      "reddit": {
        "subs": ["SMT", "ElectronicsManufacturing"],
        "api": "reddit"
      },
      "forums": [
        "https://www.eevblog.com/forum/"
      ]
    },
    "papers": {
      "arxiv": "https://export.arxiv.org/api/query?search_query={keyword}"
    }
  }
  ```

2.2 collect_news.py

- **목적**: 공식 뉴스와 기업 블로그를 수집.
- **기능**:
    1. Source_loader에서 뉴스 소스를 받아 RSS를 파싱. RSS가 없을 경우
  ```

HTML 목록 페이지를 크롤링.
2. 링크마다 _pick_first_article_url()로 실제 기사 링크를 찾고, 제목과 본문을 저장.
3. 제목/본문에 키워드가 포함되면 자료로 저장한다.

2.3 collect_papers.py

- **목적**: 학술 논문 메타데이터 수집.
- **기능**:
  1. ArXiv, IEEE Xplore 등 API/RSS를 사용하여 키워드 검색.
  2. 논문의 제목, 저자, 발표 연도, 초록, DOI를 저장.
  3. 초록 또는 제목에 키워드가 존재하는지 확인.

2.4 collect_standards.py

- **목적**: 표준/정책 문서 수집.
- **기능**:
  1. 기관 사이트에서 키워드로 검색하거나 PDF 목록 페이지를 스크래핑.
  2. 문서 제목, 출처, 발행일, 링크를 저장.
  3. 본문 내용을 수집할 필요는 없으며 링크와 간단 요약만 보유한다.

2.5 collect_community.py

- **목적**: 커뮤니티 글 수집.
- **기능**:
  1. Reddit API: 서브레딧 목록에서 키워드 검색.
  2. 기타 포럼: RSS가 있으면 RSS를 사용하고, 없으면 HTML에서 게시물 제목·조회수·댓글 수를 파싱.
  3. 추천 수, 댓글 수, 본문 길이를 고려해 필터링.

2.6 evaluate.py

- **기능**:
  1. 자료 유형별 품질 게이트(QG) 적용: 뉴스/커뮤니티 글은 본문 길이, 링크 수, 표준/정책은 신뢰도 체크.
  2. 스코어링: 도메인 신뢰도, 키워드 적합도, 시간적 중요도 등을 고려해 점수를 계산.
  3. 결과는 {id, keyword, score, type, meta, status} 형식으로 저장.

2.7 translate_summarize.py

- **기능**:
  1. 영문 뉴스/논문의 초록 및 본문에서 핵심 문장을 추출 후, OpenAI API 또는 다른 번역 API를 통해 한국어 요약 생성.
  2. 번역 API 키가 없을 경우 규칙 기반으로 요약한다.
  3. Glossary를 적용해 용어 일관성을 유지하고, 최대 2~3 문장으로 압축한다.

2.8 admin_ui.py

- **기능**:
  1. 수집된 자료 목록을 유형/점수/날짜별로 표시하고 검색·필터 기능 제공.
  2. 편집자는 각 자료에 대해 approved(예/아니오)를 선택하고 필요 시 한 줄 코멘트를 추가.
  3. 승인 결과는 DB/JSON에 저장되어 발행 단계에서 사용된다.

2.9 publish_engine.py

- **기능**:
  1. 편집자가 승인한 자료만을 모아 HTML/MD/JSON을 생성.
  2. 자료를 유형별/연대기별로 그룹화하여 키워드 역사 페이지를 구성.
  3. 최종 결과물을 archive/키워드_YYYYMMDD.html 등으로 저장하고 자동으로 게시한다.

---

3. 개발 일정 및 워크플로우

아래는 4주 단위 초기 개발 계획 예시입니다.

주 1 (설계 및 소스 준비)

- 요구사항 확정, 키워드 포맷 상세 정의.
- 소스 목록 파일(feed json) 작성.
- 모듈 구조 설계 및 Git 저장소 설정.

주 2 (수집 모듈 개발)

- RSS/HTML 크롤러(collect_news, collect_community, collect_papers, collect_standards) 개발.
- 기본 키워드 입력을 통해 테스트 수집 실행.
- 수집 결과를 JSON 형식으로 저장.

주 3 (평가·요약·번역 모듈 개발)

- evaluate.py 구현: QG, 스코어링, 신뢰도 계산.
- translate_summarize.py 구현: 핵심 문장 추출 및 번역 API 연동.
- 초기 Glossary 정의 및 적용.

주 4 (관리 UI 및 발행 엔진 개발)

- Flask/FastAPI로 간단한 관리자 UI 구축.
- publish_engine.py 구현: HTML/MD 출력 템플릿 제작.
- 전체 파이프라인 통합 및 테스트.
- 운영 매뉴얼 초안 작성.

---

4. 역할 분담 (추천)

역할	주요 업무
수집 담당	source_loader 설계, 뉴스/논문/표준/커뮤니티 크롤러 개발
평가 및 요약 담당	evaluate 모듈과 번역/요약 모듈 개발, Glossary 관리
UI 및 발행 담당	admin_ui, publish_engine 개발 및 배포 자동화
프로젝트 관리	일정 관리, 요구사항 조율, 품질 보증 및 테스트

---

5. 산출물 및 문서화

- **코드**: 각 모듈별 Python 스크립트, requirements.txt.
- **구성 파일**: feeds/official_sources.json, feeds/community_sources.json, feeds/paper_sources.json, config.json, editor_rules.json.
- **데이터베이스**: 초기 SQLite DB 또는 JSON 저장 구조.
- **운영 매뉴얼**: 설치 및 실행 방법, 키워드 입력 및 발행 절차, 문제 해결 가이드.
- **테스트 리포트**: 수집 결과 샘플, 스코어링 분포, UI 스크린샷 등.

이 개발 가이드를 따라 단계적으로 구현을 진행하면, 키워드 중심의 역사 큐레이션 뉴스 포맷을 안정적으로 구축할 수 있습니다. 향후 데이터 축적과 UI 개선을 거듭하면서 독자가 키워드에 대한 깊은 이해를 얻도록 발전시킬 수 있습니다.