퀄리저널 키워드 뉴스 발행 시스템 분석 및 개선·고도화 가이드

서론

퀄리저널(QualiJournal) 프로젝트는 “하루 하나의 키워드”를 중심으로 여러 출처의 콘텐츠를 자동으로 수집하여 기사 목록을 만들고 편집자가 승인한 최소 15 개의 기사로 뉴스 형태의 특집을 발행하는 시스템이다. 이 과정에서 수집된 모든 기사와 메타데이터는 selected_keyword_articles.json 파일에 저장되고, 편집자가 승인한 기사만 selected_articles.json 파일에 기록되어 발행에 사용된다【201301972935064†L18-L27】. 발행 전까지는 편집자가 JSON 파일을 수동으로 살펴보며 승인 표시와 코멘트를 입력해야 한다. 시스템은 Python 스크립트와 PowerShell 자동화 스크립트(run_quali_today.ps1)를 이용해 **수집→재구성→보충→동기화→발행**의 일련의 단계를 실행한다【201301972935064†L90-L103】. 발행 조건으로는 승인된 기사 수가 최소 15 개여야 한다는 품질 게이트가 있으며【201301972935064†L10-L14】, 이 조건은 features.require_editor_approval 옵션을 통해 제어할 수 있다.

본 보고서는 해당 시스템의 기술적 구조와 운영 정책을 분석하고, 최신 콘텐츠 큐레이션 트렌드와 AI 기술을 반영하여 개선 및 고도화 방향을 제시한다. 또한 실제 개발 과정에서 참고할 수 있도록 하루에 한 단계씩 완성할 수 있는 **5 단계 작업 계획**을 제안한다.

1. 현 시스템 기술 분석

1.1 파일 구조와 데이터 흐름

- **작업용 JSON 파일** – selected_keyword_articles.json 파일은 하루의 키워드에 대해 수집된 모든 기사의 제목·요약·출처·날짜·점수·승인 여부 등을 담는다【201301972935064†L18-L23】. 편집자는 이 파일을 열어 각 기사에 대해 approved 플래그와 editor_note 를 기록한다.
- **발행 대상 JSON 파일** – selected_articles.json 파일은 승인된 기사만 모은 최종 발행본이며, 발행 스크립트는 이 파일을 참고하여 Markdown/HTML 뉴스 페이지를 생성한다【201301972935064†L24-L27】.
- **아카이브** – 발행이 완료되면 해당 키워드의 페이지가 archive/YYYY-MM-DD_KEYWORD.md, .html, .json 형식으로 저장된다【201301972935064†L29-L35】. 날짜별로만 구분하므로 하루에 한 번만 발행하는 구조다【201301972935064†L260-L262】.
- **보조 스크립트** – tools/ 폴더에는 여러 파이썬 스크립트가 있다. rebuild_kw_from_today.py 는 수집된 원본 JSON 들을 표준 구조로 병합한다【201301972935064†L41-L45】. augment_from_official_pool.py 는 승인된 기사 수가 부족할 때 공식 기사 풀에서 보충한다【201301972935064†L46-L49】. force_approve_top20.py 는 상위 점수 20 개의 기사를 일괄 승인한다【201301972935064†L73-L77】. sync_selected_for_publish.py 와 repair_selection_files.py 는 승인된 기사만 발행용 JSON 으로 동기화하고 구조를 교정한다【201301972935064†L79-L88】.
- **자동화 스크립트** – Windows 환경에서는 PowerShell 스크립트 run_quali_today.ps1 를 통해 수집·재구성·보충·승인·발행 단계가 순차적으로 실행된다【201301972935064†L90-L103】.

1.2 운영 정책 및 구성

- **발행 기준** – 발행을 위해 최소 15 개의 기사가 승인되어야 한다는 품질 게이트가 설정되어 있으며, require_editor_approval 옵션으로 이 기준을 제어한다【201301972935064†L10-L14】. 승인 기사 수가 부족할 경우 응급 루틴으로 발행 시 승인 체크를 일시적으로 끌 수 있지만, 발행 후 다시 설정을 복원해야 한다.
- **커뮤니티 필터 및 점수 임계값** – 커뮤니티 출처의 잡음을 줄이기 위해 community_sources.json 과 config.json 파일에서 키워드 포함 여부, 제목 길이, 차단할 제목 패턴, 최소 추천 수/댓글 수, 점수 임계값 등을 설정한다【201301972935064†L174-L182】. 임계값을 조정하여 너무 엄격하거나 느슨하지 않도록 운영자가 튜닝한다【201301972935064†L179-L181】.
- **공식 소스 URL 관리** – official_sources.json 파일에 정의된 RSS 피드 URL 은 정기적으로 확인하고, 404 오류나 구조 변경 시 갱신해야 한다【201301972935064†L183-L188】.
- **자료 백업** – archive 폴더에 날짜별 산출물이 쌓이므로 용량 관리와 백업을 고려해야 한다【201301972935064†L258-L261】. 중복 발행 시에는 키워드_YYYYMMDD_HHMM 형태로 파일명을 조정하는 등 규칙을 마련해야 한다【201301972935064†L260-L262】.

1.3 기존 개선 제안 요약

보고서에는 이미 몇 가지 발전 방향이 제안되어 있다. 주요 내용은 다음과 같다:

- **키워드 자동 회전 및 추천 기능** – 키워드 풀을 미리 정해 자동으로 순환하거나, 전일의 뉴스 트렌드를 분석해 다음 날 키워드를 추천하는 알고리즘을 도입하여 “하루 한 키워드” 발행을 자동화할 수 있다【201301972935064†L262-L268】.
- **작업 스케줄러 연동** – 수집·발행 작업을 정해진 시간에 자동으로 실행하도록 Windows 작업 스케줄러나 cron 에 등록하고, 실행 결과를 이메일 또는 슬랙으로 알림하는 방안이 제시됐다【201301972935064†L282-L287】.
- **편집 UI 및 UX 개선** – Flask/FastAPI 백엔드와 React/Vue 프론트엔드로 관리자용 웹 UI 를 구축하여 키워드 입력, 기사 검토·승인·발행을 한 곳에서 처리할 수 있도록 제안하였다. 체크박스 승인, 편집자 코멘트 입력 칸, 상태별 필터링 기능 등을 제공하면 편집 효율이 높아질 것이다【201301972935064†L288-L297】.
- **기술 스택 고도화** – Elasticsearch 연동으로 검색 성능을 개선하고, Docker 이미지화 및 CI/CD 파이프라인을 도입해 배포를 일원화하며, Python 코드의 리팩터링과 단위 테스트 추가를 통해 안정성과 확장성을 높이는 것이 제안됐다【201301972935064†L299-L304】.

2. 시스템의 강점과 한계

2.1 강점

1. **모듈화된 파이프라인** – 수집, 재구성, 보충, 승인, 발행 단계가 각각 스크립트로 분리되어 있어 유지보수가 용이하다.
2. **품질 게이트를 통한 발행 관리** – 승인된 기사 수가 15 개 이상이어야 발행하는 정책은 뉴스 품질을 일정 수준 이상으로 유지한다【201301972935064†L10-L14】.
3. **커뮤니티 필터와 점수체계** – 커뮤니티에서 노이즈를 제거하기 위한 필터와 점수 임계값이 설정되어 있어 선택된 기사 품질을 제어할 수 있다【201301972935064†L174-L181】.
4. **비상 발행 루틴** – 승인 기사가 부족할 때 응급 모드를 통해 발행을 강행할 수 있는 안전장치가 마련되어 있다.

5. **자동화 스크립트** – PowerShell 스크립트를 통해 일련의 파이프라인을 한 번에 실행할 수 있어 운영 부담을 줄였다【201301972935064†L90-L103】.

2.2 한계 및 문제점

1. **수동 키워드 선택과 편집** – 발행자가 매일 키워드를 직접 선택해야 하고 JSON 파일을 열어 수작업으로 approved 와 editor_note 를 입력해야 한다. 이는 시간이 많이 들고 오류의 원인이 된다.
2. **제한된 사용자 인터페이스** – GUI 가 없어 편집자가 JSON 을 직접 수정해야 하므로 협업이 어렵고 실수 가능성이 높다.
3. **정적 소스 및 필터 설정** – RSS 피드 URL 과 커뮤니티 필터는 코드나 설정 파일에 고정되어 있어 변경·추가가 번거롭다. 일부 공식 소스는 URL 변경으로 404 오류를 일으키는 등 수집 실패 위험이 있다【201301972935064†L183-L188】.
4. **개발/운영 환경이 혼재** – Python 스크립트와 PowerShell 스크립트를 혼용하고 있어 플랫폼 의존성이 높고, Windows 환경에 치중돼 있다.
5. **확장성 및 개인화 부족** – 현재 시스템은 모든 독자에게 동일한 키워드 뉴스만 제공하며, AI 기반 추천이나 요약 기능이 없다. 2025 년 최신 콘텐츠 큐레이션 트렌드는 **사용자 맞춤형 콘텐츠와 AI 추천**이 핵심인데【748947534006174†L120-L134】, 현 시스템은 이를 지원하지 않는다.
6. **모니터링 및 통계 부족** – 기사 수집·발행 과정에서 성공 여부, 수집률, 조회수 등 핵심 지표를 기록·분석하는 기능이 없다.

3. 개선 및 고도화 제안

3.1 키워드 관리 및 자동화

- **키워드 자동 회전** – 보고서에서 제안한 것처럼 키워드 풀을 미리 정의하고 자동으로 로테이션하는 기능을 구현한다. 예를 들어, 주 단위로 전략적 키워드 목록을 준비하고 날짜별로 순차 선택하거나, 과거 발행 기록을 분석해 특정 키워드의 빈도와 독자 반응을 기반으로 다음 키워드를 추천한다【201301972935064†L262-L268】.
- **트렌드 기반 키워드 추천** – 최신 뉴스 트렌드를 분석하여 키워드를 추천하는 알고리즘을 개발한다. 외부 RSS 및 소셜 미디어의 검색량, Reddit 나 트위터의 관심도, Google Trends 자료 등을 수집하여 특정 키워드의 인기 변화를 평가한 뒤, 높은 관심을 보이는 키워드를 자동 선택할 수 있다. 2025 년 미디어 트렌드에 따르면, AI 추천 엔진은 독자가 관심 갖는 주제를 실시간으로 파악하고 개인화된 추천을 제공해야 유지율을 높일 수 있다【748947534006174†L120-L134】.
- **키워드 동의어 관리** – keyword_synonyms.json 파일에 정의된 동의어 목록을 활용해 “IPC-A-610”과 같이 표기법이 다른 키워드를 일괄 매칭한다. 수집 단계에서 검색어에 동의어를 포함하면 관련 기사를 놓치는 일을 줄일 수 있다.
- **스케줄러 자동화** – Windows 작업 스케줄러 또는 cron 을 활용해 매일 특정 시간에 자동으로 수집·재구성·보충·발행 과정을 실행하도록 설정한다. 성공/실패 결과는 이메일·슬랙 등으로 통지하여 운영자가 결과만 확인할 수 있게 한다【201301972935064†L282-L287】.

3.2 데이터 수집 및 필터링 고도화

- **공식 소스의 모니터링 및 자동 갱신** – official_sources.json 의 RSS 피드 URL 을 정기적으로 검사하여 HTTP 상태 코드와 구조 변화를 확인하고, 404 오류나 구조 변경 시 알림을 전

송하도록 자동화한다【201301972935064†L183-L188】. RSS 피드 갱신을 위해 별도의 스크립트를 만들어 일주일에 한 번씩 URL 목록을 점검하고, 변경 사항을 저장소에 반영한다.

- **커뮤니티 필터 튜닝** – community_sources.json 과 config.json 의 필터 임계값(추천수, 댓글수, 점수 threshold)을 기간별로 분석해 적정 수준을 자동으로 조정한다. 예를 들어 최근 일주일간 수집된 커뮤니티 글의 평균 점수와 분포를 계산하여 threshold 를 동적으로 설정한다. 필요 시 필터를 완화해 후보 기사를 늘리고, 과도한 잡음을 줄이기 위해 필터를 강화할 수 있다.
- **데이터 표준화와 오류 방지** – repair_selection_files.py 스크립트를 정기적으로 실행해 selected_articles.json 구조를 검증한다. JSON 파일을 수동으로 수정하는 대신, CLI 도구를 제공해 승인 플래그와 코멘트를 수정할 수 있도록 한다. 또한 JSON 유효성 검사를 추가해 잘못된 형식이나 인코딩 오류를 사전에 막는다.

3.3 관리자 UI 및 편집 경험 개선

콘텐츠 큐레이션 전문가들은 **타깃 독자 이해, 품질·신뢰성 유지, 가치 제공**을 핵심 원칙으로 강조한다【677409651254782†L113-L147】. 현 시스템의 편집 과정은 JSON 파일을 직접 편집해야 하므로 이러한 원칙을 실천하기 어렵다. 관리자 UI 개발을 통해 편집자 경험을 대폭 개선할 수 있다.

- **웹 기반 대시보드** – Flask 또는 FastAPI 로 REST API 를 구성하고 React/Vue 프론트엔드로 대시보드를 구축한다. 대시보드에서는 키워드 입력·수집 트리거·기사 목록 검토·승인·발행까지 일련의 과정을 브라우저 한 곳에서 처리할 수 있도록 한다【201301972935064†L288-L297】.
- **기사 승인 UI** – selected_keyword_articles.json 의 내용을 테이블로 시각화하고, 각 행에 승인 체크박스와 편집자 코멘트 입력 필드를 제공한다. 승인/보류 상태별 필터링, 기사 점수 기준 정렬, 키워드별 그룹화, 빠른 검색 기능을 추가해 편집 효율을 높인다. 코멘트 미입력 시 경고 메시지를 띄우는 UX 개선도 포함한다【201301972935064†L288-L297】.
- **히스토리 및 협업 지원** – 여러 편집자가 동시에 작업할 수 있도록 변경 이력 관리와 권한 제어를 구현한다. 승인 기록, 코멘트 수정 내역, 발행 로그 등을 저장하여 이후 트러블슈팅과 평가에 활용한다.

3.4 AI 기반 기능과 개인화

최근 콘텐츠 큐레이션에서는 **AI 추천과 맞춤형 피드**가 필수 요소로 자리 잡고 있다【748947534006174†L120-L134】. 또한 AI 는 단순히 자동화하는 것이 아니라 **인간 편집자와 협력**하는 “human-in-the-loop” 접근이 중요하다【748947534006174†L90-L110】. 이를 반영한 개선 방향은 다음과 같다.

- **요약 및 핵심 문장 추출** – AI 모델을 이용해 수집된 기사의 요약문과 핵심 키워드를 자동 추출한다. 이렇게 생성된 요약은 기사 카드에 포함시켜 독자가 빠르게 내용을 파악할 수 있게 하고, 편집자는 요약을 기반으로 승인 여부를 판단하기 쉽다.
- **개인화된 추천 엔진** – 독자의 선호도와 조회 기록을 분석해 개인화된 키워드 뉴스 또는 기사 추천을 제공한다. Recombee 보고서에 따르면, AI 추천 엔진은 사용자 행동을 분석해 관련 기사를 실시간으로 제시하며, 이메일 뉴스레터·무한 스크롤 피드 등 다양한 채널에서 개인화된 콘텐츠를 제공해야 독자를 유지할 수 있다【748947534006174†L120-L134】.
- **실시간 트렌드 분석 및 키워드 제안** – ElasticSearch, 자연어 처리(NLP), LLM 을 활용해 뉴스 전체를 스캔하고 급상승하는 주제나 키워드를 탐지한다. 트렌드 감지 알고리즘은 최근

수집된 기사에서 TF-IDF 증가율, 소셜 미디어 언급량, 기사의 독자 클릭률 등을 종합해 신속하게 키워드를 추천한다.

- **에디터와 AI 의 협업** – AI 추천을 전적으로 따르지 않고, 편집자가 결과를 수정·조정할 수 있도록 "human-in-the-loop" 방식을 채택한다. Recombee 보고서에서도 AI 는 편집자를 대신하는 것이 아니라 목소리를 강화하는 역할을 해야 한다고 강조한다【748947534006174†L90-L110】.
- **분석 및 지표 대시보드** – 추천 엔진은 사용자 행동 데이터를 실시간으로 분석하고, 관리자에게 트렌드·관심 키워드·기사별 클릭률 등 통계 정보를 제공해야 한다【748947534006174†L180-L188】. 이를 통해 키워드 선정과 기사 큐레이션 전략을 지속적으로 개선한다.

3.5 기술 스택과 운영 인프라 개선

- **모듈화 및 리팩터링** – orchestrator.py 를 여러 모듈(수집, 파싱, 필터링, 정규화, 발행)로 분리하고 함수형 스타일로 리팩터링한다. 각 모듈에 대해 단위 테스트를 작성해 안정성을 높이고, 기능 변경 시 테스트를 통해 회귀를 방지한다.
- **비동기 처리와 병렬성** – 기사 수집과 외부 API 호출은 네트워크 지연이 크므로 asyncio 나 aiohttp 를 활용한 비동기 처리로 속도를 향상시킬 수 있다. 또한 여러 RSS 피드를 병렬로 요청하는 스레드/프로세스 풀을 도입해 전체 수집 시간을 단축한다.
- **Docker 컨테이너와 CI/CD** – 서비스 전체를 Docker 이미지로 패키징하고 GitHub Actions 나 GitLab CI 를 활용해 자동 테스트 및 배포 파이프라인을 구축한다. 이를 통해 여러 환경에서 동일한 설정으로 실행할 수 있으며, 의존성 충돌을 방지한다【201301972935064†L299-L304】.
- **로그 관리와 모니터링** – 수집·발행 과정에서 발생하는 에러와 경과 시간을 구조화된 로그로 기록하고, Grafana/Prometheus 등의 모니터링 툴과 연동해 운영 현황을 시각화한다. 오류 시 알림 시스템(슬랙, 이메일)을 통해 빠르게 대응할 수 있도록 한다.
- **검색 및 분석 인프라** – Elasticsearch 또는 OpenSearch 를 도입해 수집된 기사와 메타데이터를 색인하고, 고급 검색·필터링·통계 분석을 지원한다. 자연어 쿼리와 추천 API 를 구축하면 사용자에게 더 나은 검색 경험을 제공할 수 있다【201301972935064†L299-L304】.

4. 5 단계 작업 계획

개선을 순차적으로 추진하기 위해 아래와 같이 5 개의 단계로 나누어 제시한다. 각 단계는 하루(1 일)에 집중하여 완료할 수 있는 범위로 설계되었다.

단계 1 – 환경 구성 및 코드 분석

- 기존 코드와 스크립트(orchestrator.py, tools/*, run_quali_today.ps1)를 검토하고 주요 기능을 파악한다.
- Python 가상환경을 구성하고 필요한 패키지를 requirements.txt 로 정리해 설치한다.
- 실행 환경을 Docker 컨테이너로 패키징하기 위한 기본 Dockerfile 을 작성하고, 초기 이미지 빌드와 실행을 테스트한다.
- 현재 selected_keyword_articles.json, selected_articles.json, official_sources.json, community_sources.json, keyword_synonyms.json 구조를 분석하여 데이터 모델을 설계한다.

단계 2 – 수집·필터링 고도화 및 키워드 자동화

- official_sources.json 관리 스크립트를 작성하여 각 RSS 피드 URL 의 상태를 주기적으로 검사하고, 404 또는 구조 변경 시 경고 메시지를 출력하도록 한다【201301972935064†L183-L188】.
- 커뮤니티 필터의 점수 임계값을 데이터 기반으로 조정하는 알고리즘을 구현한다. 최근 수집 결과를 분석해 적정 threshold 를 계산하고, 운영자가 손쉽게 조정할 수 있는 설정 페이지를 마련한다.
- keyword_synonyms.json 을 활용한 키워드 확장 로직을 도입하고, 키워드 풀을 주간 단위로 자동 로테이션하는 기능을 개발한다【201301972935064†L262-L268】.
- cron 또는 Windows 작업 스케줄러에 수집·발행 작업을 등록하고, 성공/실패 결과를 슬랙 또는 이메일로 통지하는 기능을 추가한다【201301972935064†L282-L287】.

단계 3 – 관리자 UI 구축 및 편집 프로세스 개선

- FastAPI 기반의 REST 백엔드를 구축하여 키워드 등록, 기사 목록 조회·승인·코멘트 입력, 발행 트리거 등의 API 를 제공한다.
- React 또는 Vue.js 를 이용해 프론트엔드 대시보드를 개발한다. 테이블 형태로 기사 목록을 표시하고, 승인 체크박스·코멘트 입력란·검색·정렬·필터 기능을 구현한다【201301972935064†L288-L297】.
- 승인/보류 상태와 편집자 코멘트의 변경 내역을 기록하고, 여러 편집자가 동시에 작업할 수 있도록 협업 기능(예: 변경 이력 표시, 잠금/경고 메시지)을 추가한다.
- JSON 파일 직접 편집을 줄이기 위해 백엔드 API 에서 데이터 검증 및 저장을 처리하도록 구현한다.

단계 4 – AI 기반 요약·추천 및 분석 기능 도입

- 수집한 기사에 대해 자연어 처리(NLP)를 적용하여 요약문과 핵심 문장을 생성하는 모듈을 개발한다. OpenAI API 또는 사내 모델을 활용할 수 있으며, 요약 결과를 프론트엔드에 표시한다.
- 사용자 행동 데이터를 저장할 수 있는 데이터베이스(예: PostgreSQL, MongoDB)를 구축하고, 독자 선호도 분석 및 개인화 추천 알고리즘을 구현한다. 추천 엔진은 독자의 클릭·조회 기록을 바탕으로 관련 기사를 제안하여 독자 유지율을 높인다【748947534006174†L120-L134】.
- AI 추천 결과에 대해 편집자가 가중치 조정이나 제외 처리를 할 수 있도록 “human-in-the-loop” 기능을 구현한다【748947534006174†L90-L110】.
- 대시보드에 통계·분석 탭을 추가하여 키워드별 관심도, 기사별 조회수, 추천 클릭률 등 지표를 시각화한다【748947534006174†L180-L188】.

단계 5 – 모듈화·배포 자동화 및 최종 테스트

- orchestrator.py 를 수집·필터링·발행·추천 등 독립 모듈로 분리하고 단위 테스트를 추가하여 안정성을 확보한다.
- 전체 서비스를 Docker 컨테이너로 패키징하고, GitHub Actions 등 CI/CD 환경을 구축해 코드 푸시 시 자동 빌드·테스트·배포가 수행되도록 한다【201301972935064†L299-L304】.
- 비동기 처리와 병렬 수집 로직을 적용하여 속도를 최적화하고, 로그/모니터링 시스템을 통해 성능 및 오류 상황을 지속적으로 관찰한다.

- 종합 테스트를 수행하여 데이터 수집·필터링·키워드 회전·편집 UI·AI 요약 및 추천 기능이 기대한 대로 동작하는지 확인한다. 이후 필요한 문서와 개발 가이드를 작성하여 프로젝트 팀에 인수인계한다.

결론

퀄리저널 프로젝트는 하나의 키워드에 집중해 양질의 뉴스를 발행한다는 명확한 목표를 가지고 있다. 기존 시스템은 모듈화된 파이프라인과 품질 게이트를 갖추고 있지만, 키워드 선정과 편집 과정의 수작업, UI 부재, AI 기능의 부재 등 개선 여지가 많다. 본 보고서에서 제시한 5 단계 계획을 따라 환경 구성, 데이터 수집·필터링 고도화, 관리자 UI 구축, AI 기반 기능 도입, 배포 자동화 및 테스트를 차례로 진행하면, 트렌드 변화에 대응하는 현대적인 키워드 뉴스 시스템으로 진화할 수 있다. 이 과정을 통해 운영자의 업무 부담을 줄이고, 독자에게 더 가치 있는 정보와 개인화된 경험을 제공하며, 시스템의 안정성과 확장성을 확보할 수 있을 것이다.