ChatGPT

퀄리저널 키워드 뉴스 발행 시스템 인수인계 보고서

전체 시스템 개요

퀄리저널(QualiJournal) 시스템은 **"하루 하나의 키워드"**를 중심으로 여러 출처의 콘텐츠를 자동 수집 및 큐레이션하여 뉴스 형태로 발행하는 것을 목표로 합니다 [1]. 편집자가 특정 키워드를 입력하면 관련된 공식 뉴스, 블로그, 학술 자료부터 커뮤니티 글까지 폭넓게 수집하고, 그 중 **최소 15건 이상의 고품질 기사 카드**를 선정(승인)하여 특별호 형태로 최종 발행합니다 [2]. 모든 수집 기사는 JSON 파일에 정리되어 저장되며, 편집자가 이를 검토해 발행할 기사를 승인한 뒤 Markdown/HTML 뉴스 페이지로 출력됩니다 [3]. **발행을 위해서는 15개 이상의 기사가 승인**(`approved=True`)되어야 하며, 이 기준은 설정 옵션 `require_editor_approval`을 통해 **엄격히 적용**됩니다 [4]. (15개 미만일 경우 발행 단계에서 오류 처리 또는 UI 상 발행 버튼 비활성화로 차단됨 [4].)

주요 파일 및 디렉토리 구조 설명

퀄리저널 프로젝트의 작업 디렉토리에는 다음과 같은 **핵심 파일들과 폴더 구조**가 존재합니다 [5] [6]:

- `data/selected_keyword_articles.json` – 하루의 키워드에 대한 **수집 결과 원본**이 저장되는 JSON 작업본입니다. 특정 키워드로 수집된 모든 기사들의 메타데이터(제목, 요약, 출처, 날짜, 점수 등)와 편집 여부가 포함됩니다 [6]. 기사별로 편집자가 승인(`approved`)했는지, 그리고 편집자 코멘트(`editor_note`)를 남겼는지도 이 파일에 반영됩니다 [7].
- `selected_articles.json` – 최종 **발행 대상 공식 선정본**을 나타내는 JSON 파일입니다. `selected_keyword_articles.json` 중 편집자가 승인한 (`approved=True`) 기사들만 추려서 이 파일에 기록됩니다. 발행 시 **이 파일의 내용만을 참조**하여 페이지를 생성하며, 프로젝트 설정에 기본 경로로 지정되어 있습니다 [8]. (커뮤니티 기사는 별도로 `selected_community.json`으로 관리됩니다.)
- `archive/YYYY-MM-DD_KEYWORD.{json|html|md}` – 발행이 완료된 **최종 산출물 아카이브**입니다. 발행 명령을 실행하면 해당 키워드의 뉴스 페이지를 Markdown (`.md`) 및 HTML (`.html`) 형식으로 생성하고, 설정에 따라 `archive/키워드_YYYYMMDD.md/html` 형태의 파일로 저장합니다 [9]. 또한 동일한 내용의 JSON (`.json`) 파일도 함께 저장하여 추후 참고할 수 있도록 합니다 (기본 설정에서 HTML/MD/JSON 모두 출력) [10]. 이로써 날짜별 키워드 특집 기록이 `archive` 폴더에 축적됩니다.
- `tools/` **디렉토리** – 워크플로우 지원을 위한 다양한 **보조 스크립트**들이 모여있는 폴더입니다. 예를 들어 승인 상태 일괄 처리, 데이터 보정, 병합 출력 등에 사용되는 파이썬 스크립트들이 포함되어 있습니다. 아래 섹션에서 각 주요 스크립트의 역할을 설명합니다.

주요 스크립트 설명

`tools/` 폴더 내 주요 Python 스크립트들과 그 기능은 다음과 같습니다:

- `rebuild_kw_from_today.py` – 당일 수집된 키워드 기사 JSON을 표준 `dict + articles` **구조로 재구성**합니다. 여러 소스에서 모인 raw 데이터를 하나의 작업본(`selected_keyword_articles.json`)으로 합치는 역할을 하며, 기사 목록을 날짜와 함께 딕셔너리 구조로 정리합니다 [11]. 이를 통해 이후 단계에서 일관된 형식으로 기사를 다룰 수 있습니다.
- `augment_from_official_pool.py` – **공식 소스 기사 풀 보강** 스크립트입니다. 현재 작업본의 기사 수나 승인된 기사 수가 부족할 경우, 미발행 상태의 공식 기사 풀에서 추가 기사들을 채워 넣어 **기사 총량을 20개 수준으로 맞추고 승인 기사도 최소 15개 이상 확보**하도록 도와줍니다 [12]. 이를 통해 매일 발행 요건을 충족하도록 **공식 기사로 부족분을 보충**합니다.

- `force_approve_top20.py` – 수집된 기사들 중 **상위 20개를 자동 승인** 처리하는 도구입니다. 점수 순 등 정렬 기준으로 상위 기사 20건의 `approved` 필드를 강제로 True로 설정하여 편집자 승인 단계를 빠르게 수행합니다. 예컨대 "--approve-keyword-top 20" 명령을 통해 상위 20건 일괄승인이 가능하며, 이 스크립트는 해당 기능을 구현합니다 [13]. (※ 실제 구현은 `orchestrator.py`의 인자 처리로 통합되었을 수 있습니다.)
- `sync_selected_for_publish.py` – 승인된 기사만으로 **발행용 데이터를 동기화**하는 스크립트입니다. `selected_keyword_articles.json`에서 `approved=True`인 기사들만 추출하여 `selected_articles.json`에 기록(갱신)함으로써, 발행 시 참고할 **최종 선택 파일**을 준비합니다 [14]. 이 과정에서 이전에 이미 승인되었던 기사들의 정보(예: editor_note 등)를 보존하여 병합합니다.
- `repair_selection_files.py` – 선택/승인 관련 JSON 파일들의 **구조를 교정**하는 스크립트입니다. 편집 과정이나 충돌 등으로 인해 `selected_articles.json` 등의 포맷이 어긋난 경우, 이를 표준적인 딕셔너리 구조(`{"date": ..., "articles": [...]}`)로 **정규화**하여 수정합니다 [15]. 발행 전에 이 스크립트를 실행해 두면 데이터 형식 오류로 인한 문제를 예방할 수 있습니다.

PowerShell 자동화 (run_quali_today.ps1)

일일 키워드 뉴스 발행 작업은 PowerShell 스크립트 `run_quali_today.ps1`으로 **원클릭 자동화**되어 있습니다. 이 스크립트는 -Keyword 파라미터로 키워드를 받아 일련의 단계들을 순차 실행합니다 (예시: `-Keyword "IPC-A-610"`). **전체 워크플로우를 자동 실행**하는 주요 단계는 다음과 같습니다 [16] [17]:

1. **수집 단계 (Collect)** – 우선 공식 소스, 커뮤니티 소스, 키워드 기반 외부 RSS 등을 모두 수집합니다. 내부적으로 `python orchestrator.py --collect` (공식 기사 수집), `--collect-community` (커뮤니티 수집), `--collect-keyword <키워드> --use-external-rss` (해당 키워드 관련 기사 추가 수집) 명령을 차례로 실행하여 **모든 후보 기사를 크롤링**합니다 [18]. (external RSS는 필요 시 Google News API 등의 외부 뉴스도 가져오는 옵션입니다.)
2. **재구성 단계 (Rebuild)** – 수집된 결과를 오늘자 키워드 작업본으로 **재구성**합니다. `rebuild_kw_from_today.py` 스크립트를 실행하여 방금 수집된 원본 JSON들을 합치고, 표준 구조의 `selected_keyword_articles.json` 파일로 **오늘의 기사 후보 목록**을 생성합니다 [19]. (기본적으로 20개 내외 기사로 구성되며, 이 단계에서 아직 approved 필드는 False로 초기화됨.)
3. **보충 단계 (Augment)** – 공식 기사 풀이 부족할 경우 추가로 **기사 채우기**를 수행합니다. `augment_from_official_pool.py` 스크립트를 실행하여 승인된 기사가 15개 미만일 때 공식 소스의 후보 중 남는 것을 작업본에 추가합니다 [12]. 이로써 **최소 15개의 기사에는 approved=True** 상태를 확보하고 전체 기사 수도 20개 안팎이 되도록 맞춥니다.
4. **승인 수 점검** – 수집 및 보충 후 **현재 승인된 기사 개수를 확인**합니다. PowerShell 내부에서 Python 원라이너를 실행하여 `selected_keyword_articles.json` 내 `approved=True`인 기사 수를 집계하고 화면에 표시합니다 [20]. 이를 통해 발행 조건(≥15개 승인)이 충족되었는지 **실시간으로 검증**합니다. (예: "현재 승인 수 = 18" 형태로 출력됨.)
5. **승인 목록 동기화** – 승인된 기사들만 **발행 대상 파일로 동기화**합니다. `repair_selection_files.py`로 포맷을 정규화한 뒤 [15], `sync_selected_for_publish.py` 스크립트를 실행하여 `selected_articles.json`에 승인 기사 목록을 갱신합니다 [14]. 이 결과 `selected_articles.json`에는 오늘 발행에 사용할 **확정 기사들만 리스트업**됩니다.
6. **발행 단계 (Publish)** – 마지막으로 **Markdown/HTML 뉴스 페이지 생성 및 저장**을 수행합니다. 기본적으로 `python orchestrator.py --publish-keyword <키워드>` 명령으로 동작하며, 앞서 준비된 `selected_articles.json`의 기사들을 불러와 Markdown 및 HTML 페이지를 생성, `archive/` 디렉토리에 파일을 저장합니다 [21]. 만약 승인된 기사 수가 15개 이상이면 **정상 모드 발행**을 진행하며, 15개 미만일 경우 **응급 루틴(emergency)**이 동작합니다 [22]. 응급 루틴에서는 `config.json`의 `require_editor_approval` 설정을 **일시적으로 false로 변경하여 게이트를 해제**한 후 발행을 강행하고 [23], 발행 완료 직후 해당 설정을 다시 true로 복구합니다 [24]. 이렇게 해서라도 출력된 경우, 발행 후에는 반

드시 설정이 원래대로 돌아오므로 다음 작업에는 영향이 없습니다. (응급 루틴은 자동화 안전장치로, 승인 기준 미달 시에도 그날 페이지 생성을 시도하는 기능입니다.)

(※ 위 PowerShell 스크립트는 작업 루트에서 실행해야 하며, `Run with PowerShell` 또는 명령줄에서 ExecutionPolicy Bypass 옵션으로 실행 가능합니다. 스크립트 실행 완료 시 Done. 메시지가 출력됩니다.)

주요 운영 정책 설명

퀄리저널 시스템에는 **원활한 운영을 위한 몇 가지 정책**과 설정이 적용되어 있습니다:

- **발행 조건 (최소 승인 수)**: 키워드 뉴스 한 호를 발행하기 위해 **최소 15개의 기사**가 편집자에 의해 승인되어야 합니다. 이 조건은 품질 게이트(QG)로 작동하여, 15개 미만 승인 시 발행 명령에서 오류가 발생하거나 UI상 발행 버튼이 비활성화되도록 설계되어 있습니다 [4]. 권장되는 선정 기사 수는 15~20개 정도로, 지나치게 많거나 적지 않도록 균형을 맞추는 것이 좋습니다 [25].
- `require_editor_approval` **설정**: `config.json` 내 `features.require_editor_approval` 값으로 위 발행 조건을 **강제할지 여부**를 제어합니다. 기본값은 True이며, 이 경우 편집자 승인이 없는 기사는 발행에서 제외되고 승인 기준 미달 시 발행을 차단합니다 [26]. (응급 상황 시 일시적으로 False로 변경하여 발행할 수 있으나, 일반적인 운영에서는 항상 True로 유지합니다.) 이 설정을 통해 **편집자 검수 없이는 발행이 불가**한 품질 게이트를 유지합니다.
- **커뮤니티 수집 필터 및 점수 임계값**: 커뮤니티 출처(Reddit, 포럼 등)에서 잡음이 많기 때문에, `community_sources.json` 및 `config.json`의 `community.filters`에서 **여러 필터 기준**을 적용합니다. 예를 들어 키워드 포함 여부(`require_keyword`), 최소 제목 길이, 차단할 제목 패턴, 최소 추천수/댓글수 등이 설정되어 있습니다 [27]. 특히 `score_threshold` **(점수 임계값)**가 적용되어, 수집된 커뮤니티 글의 종합 점수가 해당 값 이상일 때만 후보로 채택됩니다. 이 임계값은 시스템 운영 중 **튜닝 가능**하며, 현재 약 **3.5 ~ 4.2 수준**에서 조정하여 너무 엄격하거나 느슨하지 않도록 균형을 맞춥니다 [28] [29]. 필요 시 커뮤니티 결과가 너무 적게 나오면 이 값을 낮추거나 필터 조건을 완화해 결과를 늘릴 수 있습니다.
- **공식 소스 URL 관리**: `feeds/official_sources.json`에 정의된 공식 소스들의 RSS/뉴스 피드 URL들은 **정기 점검 및 업데이트**가 필요합니다. 일부 소스는 웹 구조 변경 또는 URL 변경으로 인해 **HTTP 404 오류**를 일으키거나 최신 기사를 가져오지 못하는 경우가 있습니다. 예를 들어, 특정 산업 뉴스 사이트의 피드가 폐지되었거나 iconnect007과 같은 소스 URL에 페이지 번호가 고정되어 있어 업데이트가 필요할 수 있습니다. 이러한 공식 소스 URL들은 주기적으로 확인하여 **유효한 주소로 갱신**해야 하며, 수집 모듈에서 첫 기사 추출에 실패할 경우 로그를 통해 원인을 파악하고 URL이나 파싱 로직을 개선해야 합니다. (코드에서는 HTTP 상태 코드가 200이 아닌 경우 해당 소스를 건너뛰도록 처리되어 있습니다 [30].)

에러 대응 및 자동화 보강

운영 중 발견된 오류 상황과 **자동화 개선사항**은 다음과 같습니다:

- **f-string SyntaxError 문제**: PowerShell에서 인라인 Python 코드를 실행할 때 f-string을 사용하면 중괄호 `{}` 문자가 PowerShell의 문자열 포맷으로 간주되어 **구문 오류**가 발생한 사례가 있습니다. 예를 들어 `python -c "print(f'...{variable}...')"` 형태가 문제가 되었는데, 이 경우 f-string을 회피하기 위해 **전통적인 포맷팅이나 단순 print문으로 대체**해야 했습니다. 실제로 본 스크립트에서는 Python 코드를 실행할 때 heredoc 방식(`python - << 'PY' ... PY`)을 사용해 이러한 문제를 피했습니다 [20]. 요약하면, **PowerShell에서는 f-string 사용을 지양**하거나, 필요한 경우 중괄호를 이스케이프 처리해야 합니다.
- **PowerShell 내 Python 호출 방식**: Windows PowerShell 환경에서는 Python 스크립트를 호출할 때 경로나 인코딩 이슈로 곧장 실행이 안 되는 경우가 있어, `python -c "<code>"` **형태로 호출**하는 방식을 주로 활용했습니다. 특히 간단한 작업(예: 승인 수 계산 등)은 별도 .py 파일을 실행하기보다 파이썬 한 줄 코드를 `-c` 옵션이나 heredoc으로 전달하는 편이 이식성 면에서 수월했습니다. (PowerShell에서 파이썬 멀티라인 실행 시에는 위와 같이 here-string 블록을 사용하는 것이 가독성과 안전성 면에서 좋습니다.)

- **응급 발행 루틴**: 앞서 언급한 **응급 발행 모드**는 승인된 기사 수가 부족할 때 사용됩니다. 이 루틴에서는 `require_editor_approval` 게이트를 임시로 꺼서 발행을 시도하며, 자동화 스크립트에서 이를 구현해 두었습니다 [23]. 발행 직후에는 설정 파일을 원래대로 복구하여 이후 작업에 영향이 없도록 해줍니다 [24]. 만약 응급 루틴이 트리거되었다면, **발행 완료 후 config.json의 해당 옵션이 true로 돌아갔는지** 꼭 확인해야 합니다. 이 기능은 어디까지나 비상용이므로, 근본적으로는 수집 단계에서 15개 이상 기사를 확보하거나 편집자가 수동 승인하여 응급 상황을 피하는 것이 바람직합니다.
- **기타 자동화 보강**: PowerShell 스크립트와 Python 모듈 간 연동 시 발생하는 인코딩 이슈, 경로 문제 등을 개선해왔으며, 현재 `run_quali_today.ps1`에서는 모든 단계에 `-ExecutionPolicy Bypass` 및 `Set-Location` 등을 사용해 환경을 통일했습니다. 또, Windows 환경 특성상 GUI 없이 백그라운드 실행할 경우를 대비해 로그를 남기고 (`archive/reports/` 내 로그 파일 생성) 예외 발생 시 PowerShell `$ErrorActionPreference = "Stop"`으로 중단하도록 설정해 두었습니다. 향후 오류가 발생하면 **로그 파일과 콘솔 메시지를 확인**해 문제를 진단하고, 필요 시 예외 처리를 강화해야 합니다.

유지보수 포인트

시스템을 안정적으로 유지하기 위해 유의해야 할 **관리 포인트**는 아래와 같습니다:

- **선정 파일 구조 유지**: `selected_articles.json` (및 커뮤니티용 `selected_community.json`) 파일은 항상 `{"date": "<YYYY-MM-DD>", "articles": [ ... ]}`의 **딕셔너리 구조**를 유지해야 합니다. 리스트만 있거나 키 이름이 다르게 저장되면 발행 모듈이 제대로 인식하지 못하므로 주의합니다 [31]. 스크립트들이 해당 파일을 읽고 쓸 때 `date`와 `articles` 키를 기본 전제로 동작하므로, 수동 편집 시에도 구조를 지켜야 합니다. 만약 구조가 어긋났다면 앞서 언급한 `repair_selection_files.py`로 교정 가능합니다.
- **JSON 편집 방법**: PowerShell 스크립트 내에서 JSON 설정값을 변경해야 할 경우(예: config.json에서 `require_editor_approval` 토글 등) **반드시 JSON 파싱을 통해 수정**해야 합니다. 즉, `Get-Content ... | ConvertFrom-Json`으로 불러와 객체로 편집한 후, `ConvertTo-Json`을 통해 저장하는 방식을 사용합니다 [32]. 이렇게 해야 인코딩이나 형식 문제가 없으며, 수동으로 문자열을 치환하거나 부분 수정하다가 JSON 구문 오류가 나는 일을 방지할 수 있습니다. (`run_quali_today.ps1`에서는 ConvertFrom/To-Json으로 설정을 임시 변경하는 예를 볼 수 있습니다.)
- **데이터 파일 일관성**: `data/selected_keyword_articles.json` 작업본과 `selected_articles.json` 발행본 간 **동기화 상태**를 잘 관리해야 합니다. 일반적으로 앞의 **sync 스크립트**가 이를 보장하지만, 편집자가 수동으로 JSON을 편집하여 승인 표시를 한 경우 `sync_selected_for_publish.py`를 다시 실행하여 반영하는 것을 잊지 않아야 합니다. 또한 `selected_community.json` (혹은 archive 내 저장)도 마찬가지로 커뮤니티 승인 시 갱신해주어야 합니다. 두 파일 모두 최신 승인 현황을 반영하고 있어야 추후 키워드 교체나 UI 표시에서 혼동이 없습니다.
- **파일 백업 및 아카이브**: `archive/` 폴더에 일자별 산출물이 쌓이므로 용량 관리 및 백업을 고려해야 합니다. 오래된 archive 파일은 필요 시 압축하거나 별도 저장소로 옮기고, **동일 키워드로 재발행**하는 경우 이름 충돌이 없도록 날짜 포맷(예: 중복 발행 시 키워드_YYYYMMDD_HHMM 형태로) 규칙을 지키는 것이 좋습니다. 현재 시스템은 날짜별로만 구분하므로 하루 한 번만 발행한다는 전제에서 작동합니다.

향후 개선 및 다음 단계 제안 (선택사항)

마지막으로, 시스템의 **향후 발전 방향**과 추가 가능한 개선 사항을 제안합니다:

- **키워드 자동 회전 및 추천**: 현재는 발행자가 직접 키워드를 지정하지만, 향후 **키워드 풀을 미리 정해 자동 로테이션**하거나 **인기 트렌드 키워드 추출** 기능을 도입할 수 있습니다. 예를 들어 주간 단위로 미리 정의된 키워드 목록을 순환하거나, 전일의 뉴스 트렌드를 분석해 다음 날 키워드를 추천하는 알고리즘을 통해 **자동 키워드 선택**을 구현할 수 있습니다. 이를 통해 운영자의 수동 개입 없이도 "하루 한 키워드" 발행이 지속적으로 이루어질 수 있습니다.

- **작업 스케줄러 연동**: 매일 아침 정해진 시간(예: **09:00**)에 자동으로 수집·발행 작업이 실행되도록 Windows 작업 스케줄러 또는 cron에 등록하는 것을 권장합니다 [33]. 현재 PowerShell 스크립트를 수동 실행하고 있다면, 해당 스크립트를 스케줄러에 등록하여 **무인 자동 실행**체계를 구축합니다. 실행 후 성공/실패 여부를 이메일 알림 또는 슬랙 웹훅 등으로 통지하면 더욱 좋습니다. 자동화 주기가 안정되면 운영자는 결과물만 모니터링하면 되므로 편의성과 일관성이 향상됩니다.
- **편집 UI 및 UX 개선**: 키워드 뉴스의 편집 작업을 보다 쉽게 하기 위해 **관리자용 웹 UI**를 개발하는 것이 다음 과제로 제시됩니다 [34]. 예를 들어, Flask나 FastAPI 기반의 간단한 백엔드 API를 만들고 React/Vue 등의 프론트엔드 대시보드를 구축하여, **브라우저 상에서 키워드 입력 → 수집 트리거 → 기사 목록 검토 및 선택 → 발행**까지 한 곳에서 처리하도록 할 수 있습니다 [34]. UI 상에서는 `selected_keyword_articles.json`의 내용을 테이블로 불러와 **체크박스**로 기사 승인 선택, **한줄 편집자 코멘트 입력 필드** 제공, 승인/보류 상태별 **필터링 옵션** 등을 구현하여 편집 효율을 높일 수 있습니다. 특히 editor_note 입력칸을 두어 **코멘트 미입력 시 경고나 플레이스홀더**를 표시하는 등 UX를 개선하면 편집자가 중요한 코멘트를 빠뜨리지 않도록 도울 수 있습니다 [35]. 이러한 UI 도구를 통해 여러 편집자가 동시에 작업하거나 이전 히스토리를 쉽게 조회하고, 발행도 버튼 클릭 한 번으로 수행할 수 있게 되어 **협업과 운영 효율**이 대폭 향상될 것입니다.
- **기술 스택 고도화 및 기타 개선**: 장기적으로는 검색 최적화와 대용량 데이터 대응을 위해 **Elasticsearch와의 연동**을 고려할 수 있고, Docker 이미지化 및 CI/CD 파이프라인을 도입해 배포를 일원화할 수 있습니다 [36] [37]. 또한 Python 코드 전반의 리팩토링으로 안정성과 확장성을 높이고, 에러 로그 관리 체계를 갖추며, 중요한 기능에는 단위 테스트를 추가하는 것이 권장됩니다. 최종적으로 발행된 키워드 뉴스 페이지들이 외부 독자들에게 더 잘 도달하도록 **프론트엔드 개선 및 SEO 최적화**도 고민해볼 수 있습니다 (예: 메인 웹사이트에 키워드 뉴스 섹션 신설, 검색 엔진 friendly한 메타태그 추가 등). 이러한 개선 사항들은 프로젝트의 미래 로드맵에서 우선순위에 따라 순차적으로 추진하면 될 것입니다.

  이 문서는 퀄리저널 키워드 뉴스 발행 시스템의 작업 연속성을 위한 **공식 인수인계 기록**입니다. 향후 모든 유지보수/자동화는 이 보고서를 기반으로 한다.

---

[1] [2] [3] [4] [5] [6] [7] [21] [25] [27] [35] 1004_1퀄리저널 (QualiNews Keyword News) 프로젝트 인수인계 문서.pdf
file://file_0000000067dc622fa005d7c8c93689a9

[8] [10] [13] [26] [28] [30] [31] orchestrator.py
file://file_000000007a0861f4b00882b75601f431

[9] 1004_1Areport.pdf
file://file_000000001bdc61fda513bc87e036de91

[11] [12] [14] [15] [16] [17] [18] [19] [20] [22] [23] [24] [32] run_quali_today.ps1
file://file-JFGQQGrJBDsiBtWesZfqGG

[29] community_sources.json
file://file_00000000856861f9b0f9e6fd20319899

[33] [34] [36] [37] 1004_1퀄리저널 (QualiNews Keyword News) 프로젝트 인수인계 문서.pdf
file://file_00000000319461fa971c1027ab0d0e17