📋 Troubleshooting Guide: Tag Extraction & Async Processing

> **작성자**: Jay **작성일**: 2025년 11월 10일
> **프로젝트**: FlowNote MVP - PARA Classification System
> **브랜치**: `feat/dashboard-para-api-integration`

---

1. 📌 문제 요약

1.1. 발생 일시

- **2025년 11월 10일** (약 10시간 이상 디버깅)

1.2. 핵심 문제

1. **Tag 추출 실패**: KeywordClassifier의 LLM 응답에서 `tags` 필드가 리스트가 아닌 문자열로 반환되어 파싱 오류 발생
2. **비동기 처리 문제**: `router` 엔드포인트 간 데이터 전달 오류 (동기/비동기 혼용)
3. **Router 파라미터 불일치**: `data_manager.log_classification()`에 전달되는 파라미터 개수 불일치
4. **user_context 전달 실패**: `areas` 리스트가 문자열로 변환되어 전달되지 않음

---

2. 🔥 문제 발생 배경

2.1. Tag 추출 문제

- **위치**: `backend/classifier/keyword_classifier.py`
- **원인**:
  - LLM이 JSON 응답을 반환할 때, `tags` 필드가 **리스트가 아닌 문자열**로 리턴됨
  - 예: `"tags": "코드, 품질, 관리"` (❌) → `"tags": ["코드", "품질", "관리"]` (✅)
- **증상**:

```
ERROR: 'str' object has no attribute 'append'
```

2.2. 비동기 처리 문제

- **위치**: `backend/routes/classifier_routes.py`
- **원인**:
  - `/file` 엔드포인트에서 `keyword_classifier.classify()` (동기)와 `keyword_classifier.aclassify()` (비동기)를 혼용
  - FastAPI는 비동기 컨텍스트이므로 `aclassify()` 사용 필수
- **증상**:

```
RuntimeWarning: coroutine 'KeywordClassifier.aclassify' was never
awaited
```

2.3. Router 파라미터 불일치

- **위치**: `backend/routes/classifier_routes.py` (Line 710~715)
- **원인**:
  - `data_manager.log_classification()`에 전달되는 파라미터가 7개인데, 메서드 정의는 5개만 받음

```
  # ❌ 잘못된 호출 (7개)
  csv_log_result = data_manager.log_classification(
      user_id, file_id, filename, para_category, keyword_tags, confidence,
None
  )

  # ✅ 올바른 호출 (5개)
  csv_log_result = data_manager.log_classification(
      user_id=user_id,
      file_id=file_id,
      filename=filename,
      category=para_category,
      tags=keyword_tags
  )
```

2.4. user_context 전달 실패

- **위치**: `backend/routes/classifier_routes.py` (Step 2: 프로필 로드)
- **원인**:
  - `areas`가 쉼표로 구분된 문자열로 저장되어, KeywordClassifier에 전달할 때 리스트로 변환되지 않음

```
  # ❌ 문제 코드
  areas_list = user_profile.get("areas", "")  # "코드 품질 관리,기술 역량 개발"

  # ✅ 수정 코드
  areas_list = user_profile.get("areas", "").split(",") if
user_profile.get("areas") else []
```

---

3. ✅ 해결 방법

3.1. Tag 강제 파싱 로직 추가

파일: `backend/classifier/keyword_classifier.py`
수정 위치: `aclassify()` 메서드

```
    # 🔥 강제 파싱 로직 추가
    raw_tags = result.get("tags", [])
```

```
    logger.info(f"[{self.instance_id}] 📦 Extracted tags: {raw_tags}
(type: {type(raw_tags)})")

    # ✅ 리스트 검증 및 변환
    if isinstance(raw_tags, str):
        # 문자열이면 쉼표로 분리
        tags = [tag.strip() for tag in raw_tags.split(",") if tag.strip()]
        logger.warning(f"[{self.instance_id}] ⚠ tags가 문자열이었음. 리스트로 변
환: {tags}")
    elif isinstance(raw_tags, list):
        tags = raw_tags
        logger.info(f"[{self.instance_id}] ✅ 리스트 검증 완료: {len(tags)}개")
    else:
        tags = []
        logger.error(f"[{self.instance_id}] ❌ tags 타입 오류:
{type(raw_tags)}")
```

3.2. 비동기 처리 통일

파일: backend/routes/classifier_routes.py
수정 위치: /file 엔드포인트 (Step 4: 키워드 추출)

```
    # ❌ 동기 함수 사용 (잘못된 방법)
    keyword_result = keyword_classifier.classify(text=text,
user_context=user_context)

    # ✅ 비동기 함수 사용 (올바른 방법)
    keyword_result = await keyword_classifier.aclassify(text=text,
user_context=user_context)
```

3.3. data_manager.log_classification() 파라미터 수정

파일: backend/routes/classifier_routes.py
수정 위치: Step 5 - CSV 로그 저장

```
    # ✅ 올바른 파라미터 전달 (키워드 인자 사용)
    csv_log_result = data_manager.log_classification(
        user_id=user_id,
        file_id=file_id,
        filename=filename,
        category=para_category,
        tags=keyword_tags
    )
```

파일: backend/data_manager.py
메서드 시그니처 확인:

```
def log_classification(
    self,
    user_id: str,
    file_id: str,
    filename: str,
    category: str,
    tags: list
) -> bool:
```

3.4. user_context areas 리스트 변환

파일: backend/routes/classifier_routes.py
수정 위치: Step 2 - 프로필 로드

```
    # ✅ areas를 리스트로 변환
    areas_str = user_profile.get("areas", "")
    areas_list = areas_str.split(",") if areas_str else []

    # ✅ user_context 생성
    user_context = {
        "user_id": effective_user_id,
        "file_id": file_id or file.filename,
        "occupation": user_profile.get("occupation", "일반 사용자"),
        "areas": areas_list,  # ✅ 리스트로 전달
        "interests": user_profile.get("interests", "").split(",") if
user_profile.get("interests") else [],
        "context_keywords": areas_list  # ✅ KeywordClassifier에 전달
    }
```

---

4. 📊 수정 결과

4.1. ✅ 성공 로그

```
    INFO:backend.routes.onboarding_routes:[Step1] Generated user_id:
user_2ee2b560, occupation: 개발자
    🔵 [DATA_MANAGER] 저장 시도: user_id=user_2ee2b560, occupation=개발자
    ✅ [DATA_MANAGER] 저장 완료!

    INFO:backend.routes.classifier_routes:✅ PARA 분류 완료:
    - Category: Archives
    - Confidence: 0.85

    INFO:backend.routes.classifier_routes:✅ 키워드 분류 완료:
    - Tags: ['코드', '품질', '관리', '시스템 안정성', '성능 최적화']
    - Confidence: 0.9
    - User Context Matched: True
```

```
✅ [CSV LOG] classification_log.csv 기록 완료: react_tutorial.md
✅ JSON 로그 저장: classification_20251110_162407_308.json
✅ user_context_mapping.json 저장: user_2ee2b560
```

4.2. 기록 데이터 확인 완료

- `루트/data/classifications/classification_log.csv` - 분류 데이터 기록 누적 확인 ✅
- `루트/data/context/user_context_mapping.json` - 사용자 컨텍스트 정보 기록 누적 확인✅
- `루트/data/log/`
  - 사용자 메타데이터 누적 JSON 파일 생성 ⭕
  - 누적 기록 확인 ✅
- `루트/data/users/users_profiles.csv` - 온보딩 기록 (user_id, occupation 등) 기록 누적 확인✅

---

5. 🛠️ 재발 방지 체크리스트

5.1. Tag 추출 시 타입 검증 필수

- ☐ LLM 응답에서 `tags` 필드는 **반드시 리스트**로 반환되는지 확인
- ☐ 문자열로 반환될 경우 `.split(",")` 로 강제 변환

5.2. 비동기 함수 사용 통일

- ☐ FastAPI 엔드포인트에서는 **`await` + 비동기 함수** 사용
- ☐ `keyword_classifier.aclassify()` 사용 (동기 `classify()` 사용 금지)

5.3. 파라미터 전달 시 키워드 인자 사용

- ☐ `data_manager.log_classification()`에 **키워드 인자**로 전달
- ☐ 파라미터 개수 불일치 방지

5.4. `user_context` 전달 전 타입 변환

- ☐ `areas`를 리스트로 변환 후 전달
- ☐ `interests`도 리스트로 변환 후 전달

---

6. 📝 참고 파일

파일명	위치	수정 사항
`keyword_classifier.py`	`backend/classifier/`	Tag 강제 파싱 로직 추가
`classifier_routes.py`	`backend/routes/`	비동기 처리 통일, 파라미터 수정
`data_manager.py`	`backend/`	`log_classification()` 시그니처 확인
`main.py`	`backend/`	Router 등록 확인

---

7. 💡 추가 개선 사항

1. **LLM 프롬프트 개선**: `tags` 필드가 항상 리스트로 반환되도록 프롬프트 명시
2. **Pydantic 모델 검증**: `ClassifyResponse` 모델에서 `tags` 필드를 `List[str]`로 강제
3. **로그 레벨 조정**: 디버깅 완료 후 INFO 레벨로 변경

8.🎉 최종 결과

✅ **PARA 분류 성공**: `Archives (Confidence: 85%)`

```
    # 스트림릿 화면 메타데이터
    {
        "category":"Archives",
        "confidence":0.85,
        "snapshot_id":"Snapshot(
            id='snap_20251111_012359_387963',
            timestamp=datetime.datetime(2025, 11, 11, 1, 23, 59, 388003),
            text='최신 React.js 튜토리얼을 보고 있어. 새로운 프레임워크 공부하기 좋
네.\\n',
            para_result={
                'category': 'Archives',
                'confidence': 0.85,
                'reasoning': '제공된 내용은 과거의 정보로 보이며, 현재와 관련성이 없어
Archives로 분류. 사용자 맥락이 없기 때문에 Archives로 분류하는 것이 적절함.',
                'detected_cues': [],
                'source': 'langchain',
                'has_metadata': True
                },
            keyword_result={
                'tags': ['기타'],
                'confidence': 0.0,
                'matched_keywords': {
                    'Projects': [], 'Areas': [], 'Resources': [],
'Archives': []},
                'reasoning': '사용자 맥락에 맞는 키워드가 없기 때문에 관련된 카테고리가
없음.',
                'para_hints': {'Projects': [], 'Areas': [], 'Resources':
[], 'Archives': []},
                    'actionability': 'none', 'relevance': 'none',
'user_context_matched': False, 'context_keywords': [],
'context_boost_applied': 0, 'user_areas': [], 'processing_time': '9.75s',
'instance_id': '48d4af58'},
            conflict_result={
                'final_category': 'Archives', 'para_category': 'Archives',
'keyword_tags': ['기타'],
                'confidence': 0.85,
                'confidence_gap': 0.85,
                'conflict_detected': False,
                'resolution_method': 'auto_by_confidence',
                'requires_review': False,
                'winner_source': 'para',
```

```
                'para_reasoning': '제공된 내용은 과거의 정보로 보이며, 현재와 관련성이
없어 Archives로 분류. 사용자 맥락이 없기 때문에 Archives로 분류하는 것이 적절함.',
                'reason': '명확한 승자 선택됨 (Gap: 0.85)'},
        metadata={
            'confidence': 0,
            'is_conflict': False,
            'final_category': 'Archives'})",
            "conflict_detected":false,
            "requires_review":false,
            "user_profile":{},
            "context_injected":true,
            "keyword_tags":["코드","품질","관리","시스템 안정성","성능 최적화"],
            "reasoning":"",
            "user_context":{
                "user_id":"user_2ee2b560",
                "file_id":"react_tutorial.md",
                "occupation":"개발자",
                "areas":["코드 품질 관리","기술 역량 개발","시스템 안정성 유지","API
설계 및 관리","성능 최적화"],
                "interests":[],
                "context_keywords":{
                    "코드 품질 관리":["코드 품질 관리","코드 품질 관리 관련","코드 품질
관리 업무","코드 품질 관리 프로젝트"],
                    "기술 역량 개발":["기술 역량 개발","기술 역량 개발 관련","기술 역량
개발 업무","기술 역량 개발 프로젝트"],
                    "시스템 안정성 유지":["시스템 안정성 유지","시스템 안정성 유지 관
련","시스템 안정성 유지 업무","시스템 안정성 유지 프로젝트"],
                    "API 설계 및 관리":["API 설계 및 관리","API 설계 및 관리 관
련","API 설계 및 관리 업무","API 설계 및 관리 프로젝트"],
                    "성능 최적화":["성능 최적화","성능 최적화 관련","성능 최적화 업
무","성능 최적화 프로젝트"]}},
                "user_context_matched":true,
                "user_areas":["코드 품질 관리","기술 역량 개발","시스템 안정성 유
지","API 설계 및 관리","성능 최적화"]
            }
```


✅ **키워드 분류 성공**: ['코드', '품질', '관리', '시스템 안정성', '성능 최적화'] (Confidence: 90%)
✅ **User Context 매칭 성공**: True
✅ **CSV/JSON 로그 저장 성공**
