🔧 MultiVectorRetriever API 제약 환경 트러블슈팅 완전 가이드

> **작성일:** 2025-09-29 ~ 2025-09-30
> **작성자:** Jay
> **소요 시간:** 약 10시간+
> **최종 결과:** [API 제약 우회 방법론 + 로컬 LLM 전환 가이드]

📌 목차

1. 문제 상황

🎯 학습 목표

- MultiVectorRetriever 완전 마스터
- 요약 임베딩, 가설 쿼리 생성 구현
- RAG 검색 품질 최적화

⚠️ 직면한 문제

- **API 할당량 부족**: Groq 배치 처리 1회에 대량 토큰 소비
- **배치 처리 중단**: RateLimitError 429 반복 발생
- **Function Calling 오류**: Groq API 호환성 문제
- **실수의 댓가**: 디버깅 시도만으로 할당량 소진

🌐 환경

```
- Python: 3.13.5
- LangChain: 최신 버전
- LLM 시도:
  - Groq (llama-3.3-70b-versatile): 배치 OK, 함수 NG
  - GPT-4o-mini: 배치 할당량 부족
  - Gemini-2.5-flash: 배치 할당량 부족
```

2. 시도한 방법들과 실패 원인

2.1 시도 #1: Groq 요약 생성 ❌

목표: 61개 문서 요약 생성

코드:

```
summary_chain = (
    {"doc": lambda x: x.page_content}
    | ChatPromptTemplate.from_template(
        "다음 문서를 3-4문장으로 요약하세요:\n\n{doc}"
    )
    | groq_llm
    | StrOutputParser()
)

summaries = []
for doc in split_docs:
    summary = summary_chain.invoke(doc)
    summaries.append(summary)
```

결과:

- ✅ **60/61 성공** (1m 54s)
- ❌ **마지막 문서에서 RateLimitError 429**

실패 원인:

```
Error code: 429
Rate limit reached for model 'llama-3.3-70b-versatile'
```

2.2 시도 #2: Groq Function Calling 가설 쿼리 ❌

목표: 각 문서당 5개 가설 질문 생성

코드:

```
functions = [
    {
        "name": "hypothetical_questions",
        "description": "Generate hypothetical questions",
        "parameters": {
            "type": "object",
            "properties": {
                "questions": {
                    "type": "array",
                    "items": {"type": "string"},
                    "description": "List of 5 questions"
```

```
                    }
                }
            }
        }
    ]

    groq_function = groq_llm.bind(functions=functions)
```

결과:

- ❌ **Error 400: 'functions.0.name' property missing**
- ❌ **Groq API가 OpenAI와 다른 Function Calling 스펙**

실패 원인:

- Groq의 Function Calling 형식 불일치
- 디버깅 시도마다 토큰 소비
- 결국 할당량 소진

2.3 시도 #3: 모델 혼용 전략 ⚠

고민:

- 요약: `Groq` (배치 OK)
- 가설 쿼리: `GPT-4o-mini` 또는 `Gemini`

문제점:

- ❌ **품질 일관성 우려**
- ❌ **다른 모델도 할당량 부족**
- ❌ **관리 복잡도 증가**

---

3. MultiVectorRetriever 개념 정리

💡 핵심 아이디어

문서당 여러 벡터를 생성하여 검색 품질 향상

3.1 네 가지 전략

① 작은 청크 생성

```
    목적: 정밀한 검색
    방법: 200자 child chunk + 1000자 parent chunk
    효과: 임베딩 정확도 + 맥락 보존
```

② 요약 임베딩 ⚠ (API 부족으로 미완)

```
목적: 핵심 내용 빠른 파악
방법: LLM으로 요약 생성 → 임베딩
효과: 효율성 극대화
문제: API 할당량 소진
```

③ 가설 질문 활용 ⚠ (API 부족으로 미완)

```
목적: 다양한 관점 검색
방법: 문서당 5개 가설 질문 생성
효과: 검색 포괄성 증가
문제: Function Calling 오류 + 할당량
```

④ 수동 추가 방식

```
목적: 맞춤형 검색
방법: 사용자 직접 쿼리 추가
효과: 세밀한 제어 가능
```

---

4. 부분 성공: Chunk 기반 검색

✅ 성공한 부분

구현 내용:

```python
# Child 청크 (200자) + Parent 청크 (600자)
child_splitter = RecursiveCharacterTextSplitter(chunk_size=200)
parent_splitter = RecursiveCharacterTextSplitter(chunk_size=600)

# MultiVectorRetriever 구성
retriever = MultiVectorRetriever(
    vectorstore=vectorstore,  # Child 임베딩
    docstore=store,           # Parent 원본
    id_key="doc_id"
)
```

검색 결과:

```
✅ 검색 속도: 0.2초
✅ 정확도: 66.7% (삼성 가우스 질문)
✅ 맥락 보존: 100%
```

5. 해결 방향: 로컬 LLM 전환

🚀 제안: `EXAONE 3.5` + `Ollama`

왜 로컬 LLM인가?

```
✅ API 비용: 0원
✅ 할당량: 무제한
✅ 보안: 완전 로컬
✅ 일관성: 동일 모델 사용
```

설치 방법:

```
# 1. Ollama 설치
brew install ollama  # Mac
# or https://ollama.com/download

# 2. EXAONE 3.5 다운로드
ollama pull exaone3.5:7.8b

# 3. LangChain 통합
pip install langchain-ollama
```

코드 예시:

```
from langchain_ollama import OllamaLLM

llm = OllamaLLM(model="exaone3.5:7.8b", temperature=0)

# 요약 생성 (무제한!)
summary_chain = (
    {"doc": lambda x: x.page_content}
    | ChatPromptTemplate.from_template(
        "다음 문서를 요약하세요:\n\n{doc}"
    )
    | llm
    | StrOutputParser()
)

# 가설 쿼리 생성 (무제한!)
hypothetical_chain = (
    {"doc": lambda x: x.page_content}
    | ChatPromptTemplate.from_template(
        "다음 문서에 대한 5개 질문:\n\n{doc}"
    )
    | llm
```

```
            | StrOutputParser()
        )
```

---

6. 교훈과 인사이트

💎 배운 것들

6.1 기술적 교훈

```
✅ API 의존성 위험성 체감
✅ 로컬 LLM의 필요성 인식
✅ 에러 핸들링의 중요성
✅ 배치 처리 전략 수립
```

6.2 문제해결 교훈

```
✅ 완벽을 추구하다 멈추지 않기
✅ 부분 성공도 가치 인정
✅ 대안 솔루션 탐색 능력
✅ 트러블슈팅 문서화 습관
```

6.3 실무적 인사이트

```
📌 무료 API의 한계
📌 프로덕션 환경 고려 사항
📌 비용 vs 품질 트레이드오프
📌 로컬 vs 클라우드 LLM 선택
```

---

7. 향후 계획

✅ 단기 계획

1. EXAONE 3.5 Ollama 설치
2. 요약 생성 재도전
3. 가설 쿼리 생성 완성
4. 완전한 MultiVectorRetriever 구현

🚀 장기 목표

1. 로컬 LLM 활용 가이드 작성
2. API vs 로컬 비교 문서 작성

3. 비용 최적화 전략 정리

📚 참고 자료

관련 파일

- `07_MultiVectorRetriever_1.pdf`: 첫 시도 (요약 실패)
- `07_MultiVectorRetriever_2.pdf`: 둘째 시도 (함수 오류)
- `07_MultiVectorRetriever_3.pdf`: 부분 성공 (Chunk만) → **`07_MultiVectorRetriever_1.ipynb`** 로 업로드

외부 링크

- Groq API Docs
- Ollama 설치
- EXAONE 3.5