

안녕하세요. 늘 배우고 도  
새로운 기술을 빠르게 익히  
다.

전화번호 | 010-4107-3358  
이메일 | minseo2399@naver.com  
지원분야 | Back-End 개발

지하고 과정

했습니다. 또한, KNN 알고리즘을 통해 실험하여 실질적인 성능을 검증하고 이해하고 심지어 전용한 솔루션을 개발하는 데에 기여하였습니다.

**중심의 개발자입니다**

의견을 나누고, 동의하기 어려운  
내의 관점과 비교해 어떤 점이  
활한 관계를 유지하며 마무리

ANSWER

oBae

# Framework Library

ETC REST API, OAuth 2.0, Kafka, Jsoup, Ope

**Java**

- Java 기반 Spring Boot 환경에서 RESTful API 서버를 설계 및 구현한 경험이 있습니다.  
Java 8 이상의 기능(Stream, Lambda 등)을 활용하여 효율적인 데이터 처리 로직을 작성할 수 있습니다.
- JPA를 활용한 ORM 설계와 쿼리 최적화 경험이 있으며, Spring Security와 JWT를 이용한 인증/인가 로직도 구현한 경험이 있습니다.
- 대용량 데이터 처리 및 정기 작업 자동화를 위해 Spring Batch를 활용한 배치 처리 시스템을 설계하였고, Partitioner, TaskExecutor 등을 적용하여 병렬 처리 및 성능 개선을 구현한 경험이 있습니다.

**Redis 활용**

- Redis를 활용해 채팅방 세션 관리 및 채팅방 목록 캐싱 시스템을 구현한 경험이 있습니다.
- 다중 서버 환경에서 Redis 분산 락(Redis Lock)을 적용하여 작업 중복 실행을 방지한 경험이 있습니다.

- Elasticsearch에 임베딩 벡터를  
구현한 경험이 있습니다.

- Cosine Similarity와 KNN 알고리즘을 비교·분석한 후 성능을 기준으로 설계하였습니다.
- 텍스트 기반 검색 외에도 벡터 유사도 기반 검색을 통해 상호명 추천 기능을 활용한 경험이 있습니다.

**RDBMS 및 Spring Data JPA**

- Spring Data JPA와 연동하여 MySQL 기반의 데이터 모델링 및 CRUD API를 구현한 경험이 있습니다.
- 쿼리 튜닝, 인덱스 최적화 등을 통해 성능을 개선한 경험이 있습니다.
- 정기적으로 대용량 CSV 데이터를 수집·정제하여 MySQL에 저장하는 배치 시스템을 구축한 경험이 있습니다.
- ERD 설계 및 관계형 데이터 정규화 작업도 경험하였습니다.

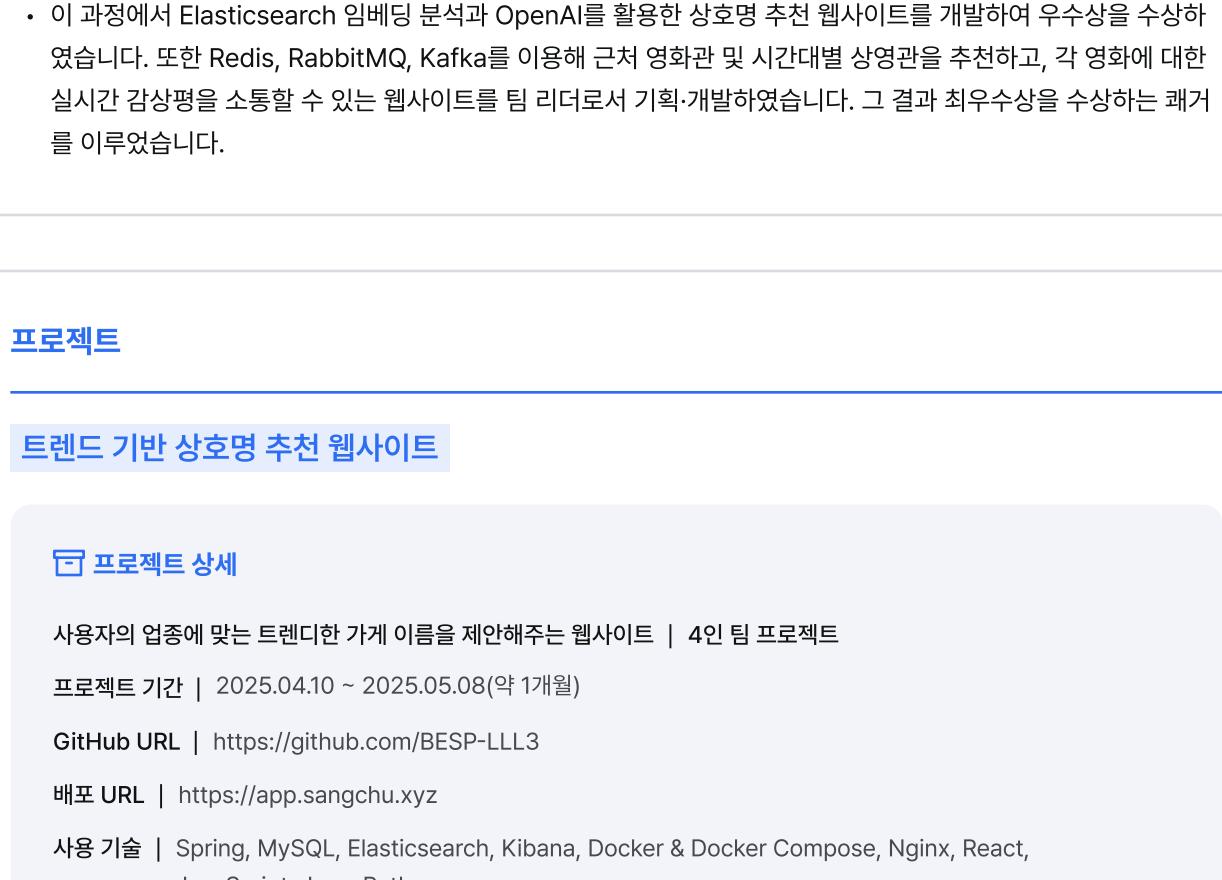
**프론트엔드 개발 및 API 역할**

- 백엔드 API 명세에 따라 React와 JavaScript를 활용하여 프론트엔드 페이지를 단독으로 구현한 경험이 있습니다.
- RESTful API를 연동하여 영화 검색, 키워드 추천, 사용자 상호작용 등의 기능을 UI로 구성하였습니다.

- Git을 활용한 버전 관리 및 협업 경  
Git Flow, Feature/Develop 브anching
- PR(Pull Request) 기반의 코드 리뷰

- 가
- Ja

## 백엔드 부트캠프 플러스 4기



## 문제

1. '삼겹살'과 같은 상호명으로 추

- ## 해결 방법

  1. 형태소 분석 시 조사, 어미, 수식어 등 불용어 품사와 한 글자 단어를 제거하여 트렌드 키워드의 정확도를 높임.
  2. 단순 키워드 나열이 아닌, 자연어 문장 형식의 강조 구문으로 상호명 + 업종 중분류 + 소분류를 조합하여 임베딩함으로써 군집 형성의 정확도와 문맥 이해도 향상.

## 성과

테스트 결과, 사용자 입력 키워드에 대해 더 현실적이고 매력적인 상호명 추천이 가능해졌으며, 추천된 상호명 품질에 대한 사용자 만족도 향상.

사용 기술 | Elasticsearch, Spring Boot, Stream API, Java Filter, Nori 형태소 분석기, HuggingFace Embedding API(in Python Flask Server), OpenAI Chat API

## 유사도 알고리즘 테스트 및 성능 최적화

### 문제

  1. Elasticsearch의 기본 search API는 최대 10,000건 제한으로 인해 전체 450만 건 데이터 기반 상호명 추천 서비스에 적합하지 않음.
  2. search\_after 방식 사용 시에도 9개 분기 약 450만 건 데이터를 순차로 불러오며 JVM의 OOM (Out Of Memory) 발생.
  3. 필터링 기준을 설정할 필요가 있었고, 이에 따라 KNN(K-Nearest Neighbors)과 Cosine Similarity를 비교 테스트하여 유사도 기반 필터링 성능과 정확도를 평가.

2.  
3.  
4.

	2020.01	2020.02	2020.03	2020.04	2020.05	2020.06	2020.07	2020.08	2020.09	2020.10	2020.11
사무소	3617.6	3550.58	3535.72	3527.5	3523.49	3500.86	3515.8	3501.52	3497.41		
증개사	3431.8	3366.35	3355.42	3352.81	3345.72	3337.72	3346.6	3332.47	3327.85		
공인	2975.2	2929.22	2919.87	2915.78	2916.38	2908.75	2923.16	2902.76	2902.03		
부동산	1875.2	1855.8	1853.32	1860.49	1851.04	1786.62	1797.29	1846.11	1845.53		
증개	625.33	637.54	641.07	648.76	648.51	637.19	638.2	642.78	650.4		
법인	230.59	238.06	240.84	244.61	239.4	242.56	236.95	247.33	250.59		
강남	145.36	144.89	144.41	146.82	143.89	133.41	133.85	140.97	146.14		
서울	51.08	91.13	90.65	90.19	86.96	87.33	85.96	83.15	83.15		
개발	80.67	88.61	88.12	87.1	85.2	87.92	83.7	83.98	80.69		
행복	55.17	76.72	74.38	74.85	75.35	72.06	71.61	73.57	73.62		
행운	66.02	63.23	62.29	62.31	60.9	59.03	58.56	62.77	60.9		
드림	66.79	63.55	61.23	60.75	60.75	61.15	61.66	60.89	59.96		
하우스	39.67	40.56	40.1	43.86	47.11	45.65	48.01	56.53	57.93		
횡금	64.5	57.86	57.38	57.87	56.47	53.19	53.2	55.62	54.2		
빌딩	43.78	53.64	54.09	56	56.93	56	55.98	52.88	52.42		

KNN 알고리즘 키워드 추출 결과 : 검색 키워드 : 부동산, k = 40, numCandidates= 300 소요 시간 약 3 분									
Keyword	202303	202306	202309	202312	202403	202406	202409	202412	202503
부동산	3.0179	4.5326	3.0227	3.0245	3.7803	4.5299	4.5382	3.7808	4.5332
사무소	2.2631	4.5304	3.7759	3.0209	3.7768	3.7721	3.7768	2.2639	4.5297
중개사	2.2631	3.7745	3.7759	3.0209	3.0209	3.7721	3.021	2.2639	3.7739
중개	2.2644	1.5133	0.7574	1.5144	2.2703	1.5144	3.0282	2.2723	2.2703
공인	2.2631	3.7745	3.7759	3.0209	3.0209	3.7721	2.2665	0.7554	2.2653
연지	0	0	0	0	0	0	0	0.7594	0.7594
아파트	0	0	0	0	0	0	0	0.7594	0.7594
구역	0.7546	0.7546	0.7546	0.7546	0.7546	0	0	1.5141	0.7594
임대	0	0	0	0	0	0	0	0.7594	0.7594

하우스	0.7574	0.7574	0.7574	0.7574	0.7574	0.7574	0.7574	0.7574	0.7574
남점	0.7574	0.7574	0.7574	0.7574	0.7574	0.7574	0.7574	0.7574	0.7574
꼬마	0.757	0.757	0.757	0.757	0.757	0	0	0	0
빌딩	0.757	2.2663	1.5117	0.757	2.2663	2.2663	1.5093	1.5093	1.5093
와룡	0.757	0.757	0.757	0.757	0.757	0	0	0	0
스테이트	0.7559	0	0.7559	0	0.7559	0	0	0.7559	0
북아현	0.7559	0	0.7559	0	0.7559	0	0	0.7559	0
팰리스	0.7555	0	0.7555	0.7555	0.7555	0.7555	0.7555	0.7555	0.7555
남산	0.7555	0	0.7555	1.5089	0.7555	0.7555	0.7555	1.5089	
트라	0.7555	0	0.7555	0.7555	0.7555	0.7555	0.7555	0.7555	0.7555
서울	0.7545	0.7545	0.7545	0	0	0	0	0	0
노른	0.7545	0.7545	0.7545	0	0	0	0	0	0
자공	0.7545	0.7545	0.7545	0	0	0	0	0	0
이상	0.7544	0	0	0	0.7544	0.7544	0	0	0

강북구	0.7544	0	0	0	0.7544	0.7544	0	0	0
개발	0.7544	1.509	0.7544	1.5111	1.5114	1.5114	2.2658	1.509	0.7546
개소	0.7544	0.7544	0.7544	0.7544	0	0.7544	0.7544	0	0
용산	0.7544	0.7544	0.7544	0.7544	0	0.7544	0.7544	0	0

## 해결 방법

1. Spring Batch 가능하도록

- 성과

  1. 배치 처리 최적화 성능 개선  
개선 전: 32.69초 → 개선 후: 10.45초 (약 69% 성능 개선)
  2. 비동기 병렬 처리 적용 성능 개선  
개선 전: 71초 → 개선 후: 52.7초 (약 26% 성능 개선)
  3. 최종적으로 450만 건 임베딩 작업을 약 5초 수준으로 최적화

사용 기술 | Spring Batch, TaskExecutor, Partitioner, HuggingFace Embedding API, Java, MySQL

# 프로젝트 기 GitHub UP 사용 기술 |

- 팀장으로 전체적인 회의 및 일정 관리 담당
- Kafka 설정, Kafka로 채팅방 별로 토픽 생성, 채팅 로그 수집 로직 개발
- Redis를 활용하여 채팅방별 사용자 목록 관리(입장, 퇴장 처리) 구현
- KMDB, KOBIS API에서 필요한 데이터 처리 및 수집 로직 개발
- React와 RESTful API 연동 프론트엔드 개발

 [Docker 사용](#)

## 서버 수평 확장 시 동시 실행 문제 해결

### 문제 상황

1. 서버 부하 분산을 위해 동일한 Spring 서버 인스턴스를 Docker 환경에서 두 개 수평 확장하여 운영.
2. 서버 기동 시 실행되는 영화 데이터 수집 로직에서,  
Elasticsearch 인덱스를 생성하는 작업이 두 인스턴스에서 동시에 실행됨.
3. 이로 인해 동일한 영화 데이터가 두 인스턴스에 동시에 충돌 및 업데이트됨.

**기술적 원인**

1. Elasticsearch는 동일한 이름의 인덱스를 여러 번 생성할 수 없기 때문에,  
두 인스턴스에서 동시에 인덱스 생성 요청이 발생할 경우 충돌 발생.
2. 이로 인해 일부 인스턴스에서 수집 로직이 실패하거나 서버 자체가 비정상 종료되는 문제가 발생.

**성과**

1. Redis 기반 분산 Lock을 적용하여,