Flicker 참고서

[목 차]

1. 협업 기반 필터링

A. Fassi 기반 이웃 탐색 알고리즘

비교 항목	FAISS 이웃 근사	브루트 포스 이웃 근사
탐색 방식	근사치 탐색 (벡터 양자화, LSH 등 사용)	정확한 탐색 (모든 벡터 비교)
시간 복잡도	$O(n \log n) \sim O(n)$	$O(n^2 \times d)$
n: 데이터 포인트 수	대규모 데이터에 적합	데이터가 많을수록 비효율적
d: 벡터 차원 수	차원 수와 상관 없이 효율적	차원이 높을수록 탐색 속도 저하
메모리 사용량	상대적으로 낮음	상대적으로 높음
정확도	근사치 제공 (정확도와 속도 균형 조정 가능)	모든 데이터에 대해 정확한 결과 제공
장점	매우 빠른 검색 속도, 대용량 데이터 처리	작은 데이터셋에서 정확한 결과
단점	정확도가 낮을 수 있음	대규모 데이터에서 속도가 느림

표 1

표 1 은 FASSI 이웃 근사와 완전 탐색 이웃 근사의 비교를 나타냄

완전탐색 기반 이웃 탐색의 경우

130,000 * 130,000 * 5 번이 소요됨 (5-> 압축된 차원의 수)

int형 (4byte) 338GB의 자원 필요로함

FASSI 기반의 거리 근사 이웃 탐색으로 변경

B. PCA (차원축소)

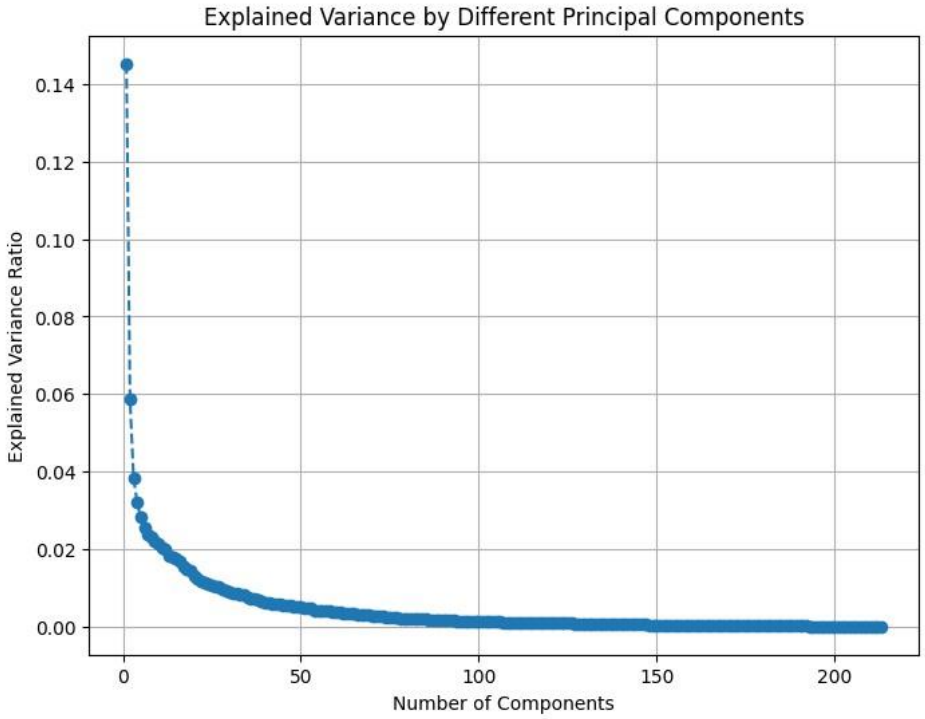

그림 1

그림 1은 x 축은 축소 차원의 수 y축은 변동 분산을 나타냄

Elbow – point 는 차원이 5 (d=5) 로 축소되었을 때로 보임

기본 데이터 설명력은 99.96% 이상으로 보임

데이터 설명력이 85% 구간 이상에서의 차원 수 설정은

전통 통계학적 관점에서 모집단을 대표하는 유의미성을 띔.

최적의 수인 d=5에서 차원 축소를 결정

13,000 -> 5 차원 축소 결정

C. KoBert Model (감성분석 모델)

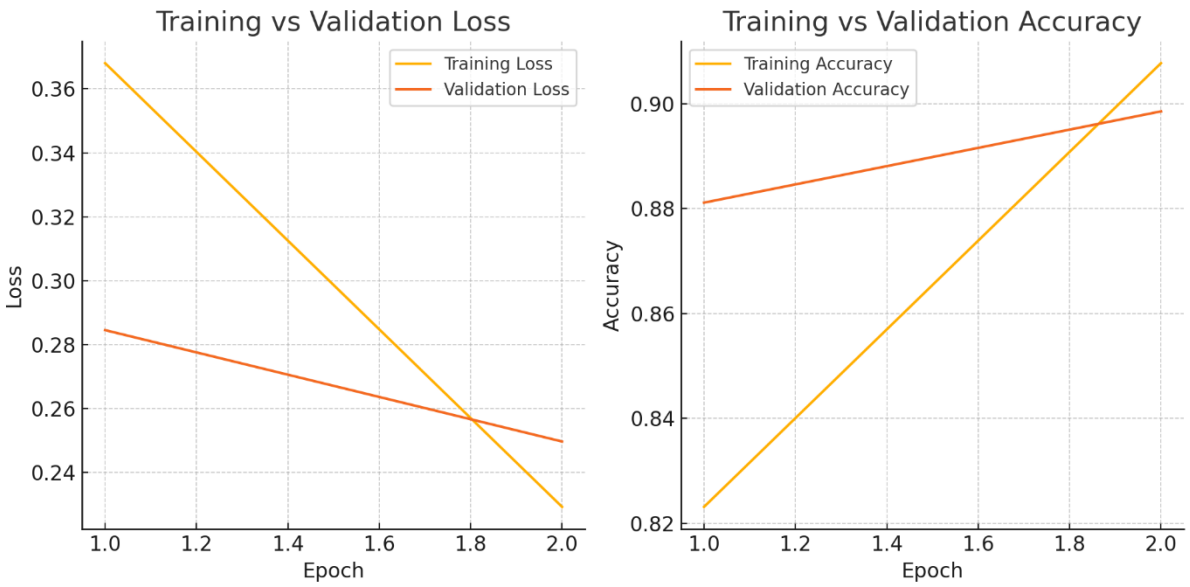

그림 2

그림 2는 Training의 Epoch에 따른 Loss(손실)과 Accuracy(정확도) 를 나타냄

정밀도 및 재현율(Class 0 부정, Class 1 긍정)

	precision	recall	f1-score	support
0	0.90	0.89	0.90	24827
1	0.90	0.90	0.90	25173
accuracy			0.90	50000
macro avg	0.90	0.90	0.90	50000
weighted avg	0.90	0.90	0.90	50000

표 2

표2는 각 수치에 대한 평가 지표에 대한 설명을 나타냄.

Precision(정밀도): 모델이 예측한 긍정 클래스(1) 중 실제로 긍정 클래스인 비율.

즉, True Positive / (True Positive + False Positive).

0.90이라는 값은 모델이 긍정 클래스라고 예측한

샘플 중 90%가 실제로 긍정 클래스라는 것을 의미

Recall(재현율): 실제 긍정 클래스(1) 중 모델이 올바르게 예측한 비율.

즉, True Positive / (True Positive + False Negative).

0.90이라는 값은 실제 긍정 클래스인 샘플 중 90%를 모델이 올바르게 예측했다는 것을 의미

F1-score: Precision과 Recall의 조화 평균으로, 두 지표 간의 균형

0.90이라는 값은 모델이 Precision과 Recall 모두에서 고르게 성능을 보였음을 의미.

Support: 각 클래스별 샘플 수

클래스 0은 24827개, 클래스 1은 25173개의 샘플

Accuracy(정확도): 전체 샘플 중 올바르게 예측한 샘플의 비율. 0.90으로, 전체 50000개의 샘플 중 90%를 정확하게 예측했다는 것을 의미

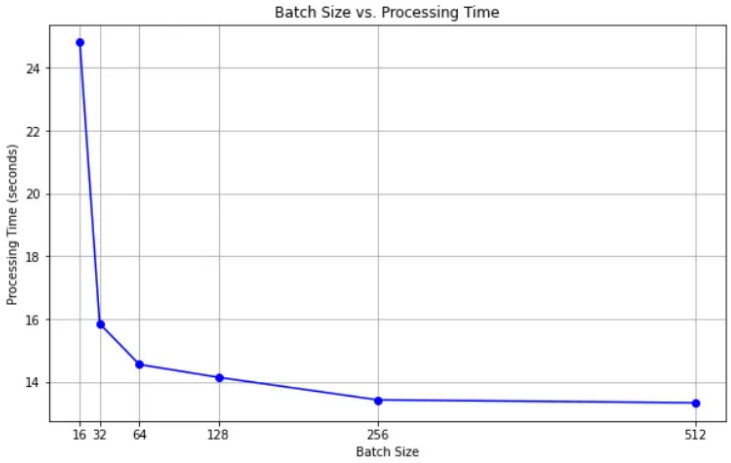

그림 3

배치 사이즈가 증가할수록 실행시간이 줄어듬

특히 batch-size가 32, 64 구간에서 눈에 띄는 감소 양상을 보임

이에 batch-size에 대한 추가적인 검증을 통해 최적의 batch-size를 조사함.

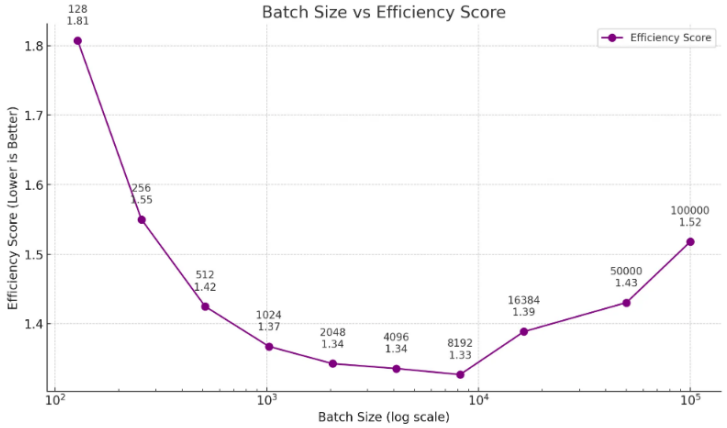

그림 4

y축은 메모리/batch-size 값의 정규화 값을 나타냄

batch-size가 1024 ~ 8192에서 low-curve가 나타남.

특히 8192 에서 이후 상승하는 Elbow-point가 관측 가능했음

이에 최적의 batch-size를 8192로 설정 후 모델 학습 진행

2. 컨텐츠 기반

A. word2vec

word2vec 훈련의 경우, 영화제목, 배우, 장르를 넣었을 때 추천 결과가 좋았고 epoch는 250에서 가장 좋은 결과를 출력.

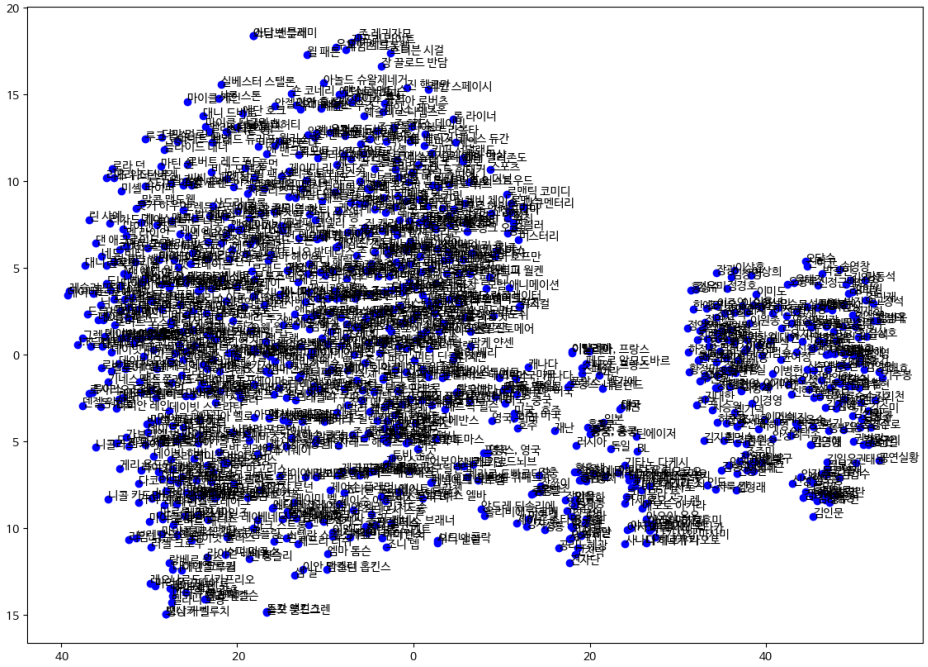

그림 5

그림 5는 상위 1000개에 대한 단어들의 분포를 나타냄.

3. MSA & INFRA

그림 6

그림 6은 Pod 수에 따른 초당 요청 건수와 평균 응답시간을 나타냄

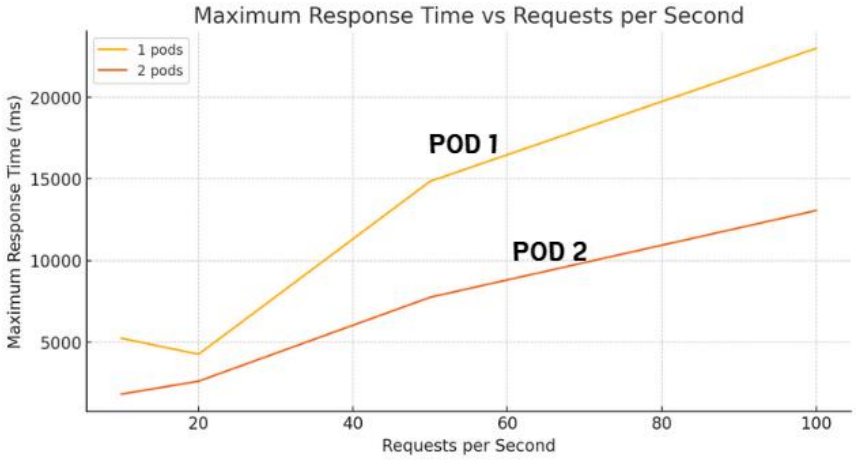

그림 7

그림 7은 Pod 수에 따른 초당 요청 건수와 최대 응답시간을 나타냄

초당 건수가 증가할수록 POD가 2개인 app의 처리 속도가 빠름

4. DB time line

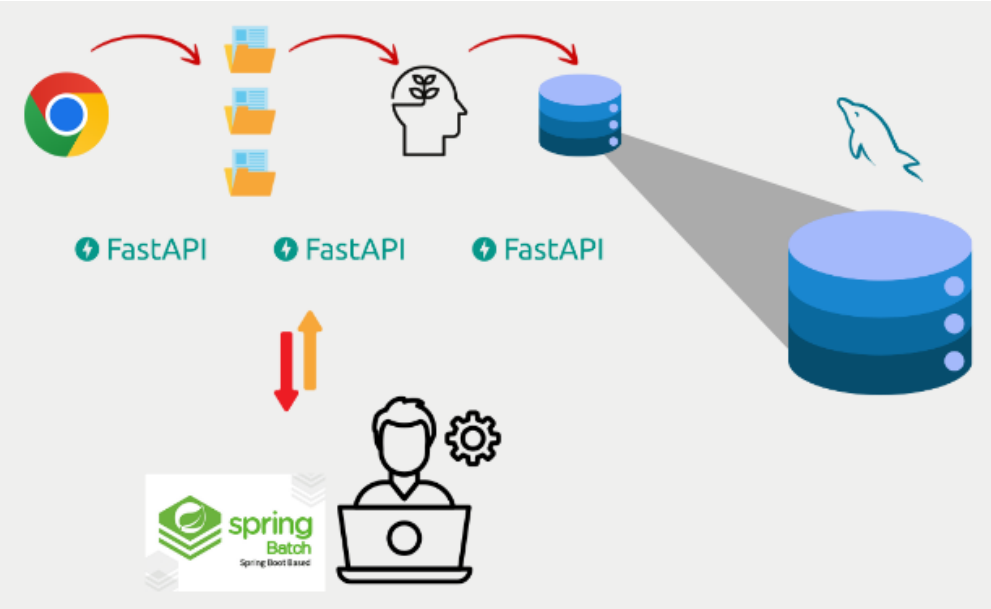

그림 8

그림 8의 경우 데이터 적재 , DB 반영까지의 FastAPI, Spring-Batch의 역할을 나타냄

현재, 하루, 일주일 단위로 계속해서 새로운 영화가 스크린 업함.

이에 따른 리뷰 데이터 급증, 반영 및 고착화 방지가 추천 서비스의 주요 목적이됨

이에 FAST API를 통해 자동 크롤링을 완성하고 ML,DL 학습 후 혹은 Law한 형태로 해당 DB 저장 로직을 완성

특히 Spring-batch가 데이터들의 변동 주기를 설정해줌.

그림 9

현재 spring-batch의 설정상 위와 같은 설정이 디폴트임.