

딥러닝을 딥하지 않게 사용해서 자연어 검색 만들기

Langcon 2023 고석현 - Toss

저는

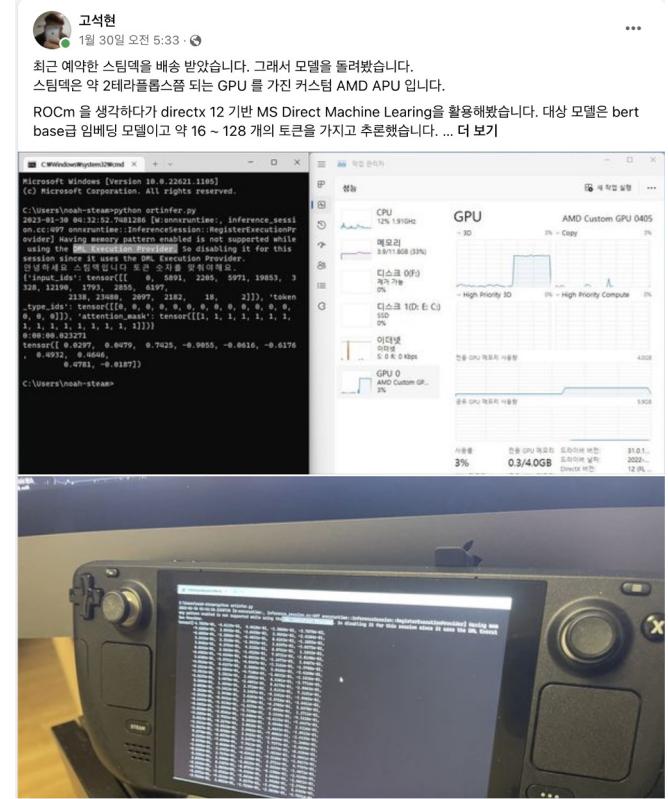


- Data Lab
이미지, 검색, 추천

Company.AI - Co-Founder
자연어 대화엔진
Naver에 M&A로 exit

Naver Clova^炬 - AI Software Platform
Hyper Clova - GPT3 inference optimization
Naver Cloud - 대화엔진 SaaS

toss - ML Platform Team
금융 AI



관심사는

챗봇 1만 개의 모델 서빙하기: AI 서비스 어디까지 해봤니

Hyper clova 거대 모델 기반의 ML 서빙과 성능 최적화 (GPT-3)

케어콜 - 코로나로부터 우리를 지키는 믿음직한 AiCall

물 흐르듯 자연스러운 ML 서비스 만들기 - MLOps 를 넘어 서비스로

성능 최적화와 ~~돈이 되는~~ 서비스 타당한 머신 러닝에 관심

딥러닝은 무엇일까요

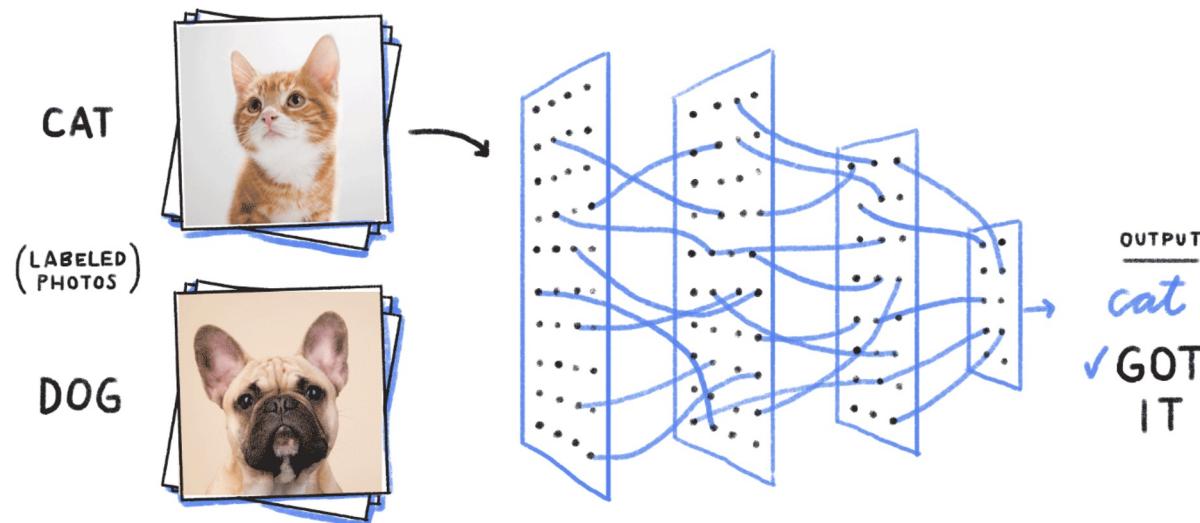
복수의 층으로 구성된 NN를 사용하는 모델과 해당하는 방식들

딥러닝은 무엇일까요

비선형 데이터의 집합에 선형성을 부여해주는 projection matrix의 학습 방식

딥러닝은 무엇일까요

feature의 정의를 스스로 하는 모델



딥러닝은 무엇일까요

지금은 Open AI로 이동한 전 Tesla Head of AI 의 Andrej Karpathy 표현을 빌려보면
새로운 개발 패러다임에 적합한 도구 -> 양질의 데이터는 코드를 일부 대체 하는 수단



Andrej Karpathy

Nov 12, 2017 · 9 min read · Listen



Software 2.0

I sometimes see people refer to neural networks as just “another tool in your machine learning toolbox”. They have some pros and cons, they work here or there, and sometimes you can use them to win Kaggle competitions. Unfortunately, this interpretation completely misses the forest for the trees. Neural networks are not just another classifier, they represent the beginning of a fundamental shift in how we develop software. They are Software 2.0.

검색은 무엇일까요

(IR) Information Retrieval ?

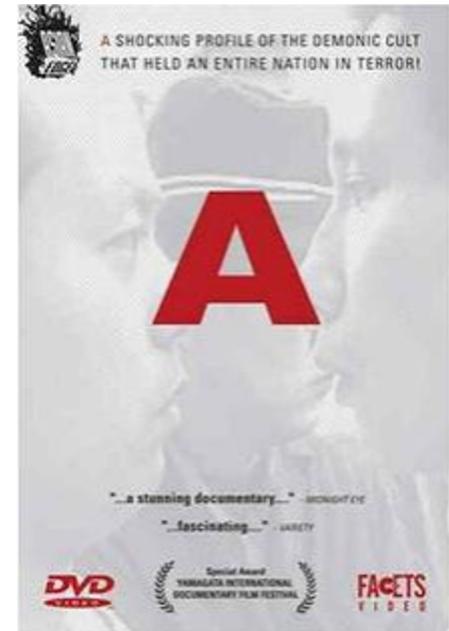
(내 의도와) 적합한 정보를 골라내는것

KR

a라는 영화 알아?



제가 알아듣기 어려운 질문 같습니다. "a"라는 제목의 영화가 무엇인지 더 자세한 정보를 제공해 주시면 대답해 드릴 수 있을 것 같습니다.



검색은 무엇일까요

정보가 (지나치게 많아서) 오염되거나 유저에게 적합한 결과가 아니라면 좋은 검색이 아닌 것

KR

You know the movie named A?



I'm sorry, but I would need more information to provide an accurate answer. There are many movies with titles that start with the letter A, and without additional context or details, I'm not sure which movie you are referring to. Could you please provide more information or clarify your question?



KR

movie name is "A"



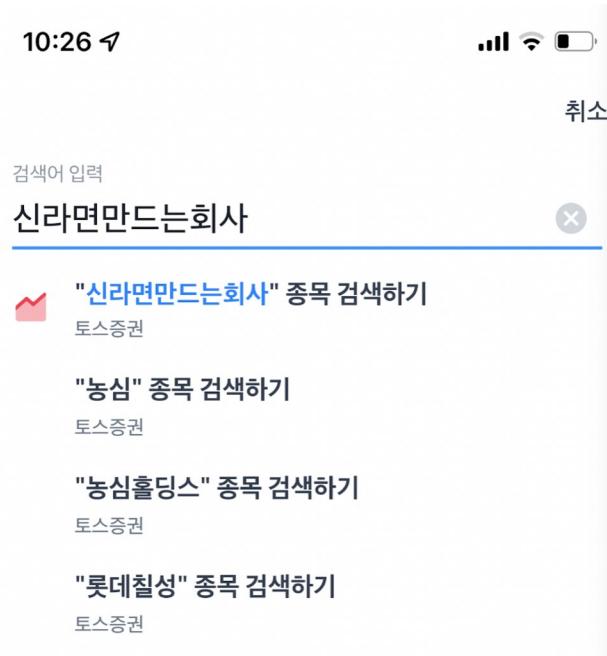
I'm still unsure which movie you are referring to, as there are many movies with a one-word title "A." Could you provide any additional information about the movie such as the director, main actors, or a brief plot summary? This would help me to better understand which movie you are asking about.



검색은 결국 유저의 의도를 자연어로 입력 받는 추천

검색을 잘하는 방법은 정말 많지만

조금만 열심히 하고 잘하고 싶었습니다.



검색 잘 하는 것은 충분히 어려우며 동시에 방대한 분야

항상 유저들은 업계 최고 수준의 엔진과 포탈의 검색 결과를 익숙하게 사용
즉 유저들의 검색에 대한 기대치는 매우 높음

대규모의 인프라와 검색 팀을 가지고 있지 않다면 좋은 검색 만들고 운영하는 건
어려운 과제

하지만 LM과 LLM은 비교적 소규모 검색을 쉽고 빠르고 만들 수 있는 강력한 도구

그래서 쉬운 딥러닝을 써서 조금만 열심히 하고 잘하고 싶었습니다.

소프트웨어 2.0

나는 때때로 사람들이 뉴럴넷을 "기계학습 도구 상자의 또 다른 도구"로 언급하는 것을 본다. 뉴럴넷은 장단점이 있다. 여기 저기에서 동작하고, 때로는 캐글 경쟁에서 승리하는데 사용할 수도 있다. 그러나 이 해석은 완전히 나무만 보고 숲을 보지 못하는 것이다. 뉴럴넷은 단지 또 다른 종류의 분류기가 아니다. 그것은 우리가 소프트웨어를 작성하는데 있어 근본적인 변화의 시작을 나타낸다. 그것은 소프트웨어 2.0이다.

쉽게 이용할 수 있다.

나는 '딥러닝은 얄팍하다'라는 농담을 좋아한다. 이것은 당신이 유용한 무언가를 얻기 위해 박사 학위가 필요하지 않다는 것이다. 기본 개념은 기초 선형 대수, 미적분, 파이썬 및 CS231n의 일부 강의 정도이다. 물론 오랜 경험에서 얻을 수 있는 전문 지식과 직관력의 양은 아주 많다. 보다 정확한 말은 소프트웨어 2.0 스택은 쉽게 이용할 수 있지만, 쉽게 마스터 할 수는 없다는 것이라.

그것은 당신보다 낫다.

마지막으로, 그리고 가장 중요하게, 뉴럴 네트워크는 나나 당신이 현재 가치있는 업종의 상당 부분에서 만들 수 있는 것보다 더 훌륭한 코드라는 것이다. 그것은 현재의 이미지 / 비디오, 음성 / 담화와 별로 관련이 없다.

조금만 열심히 하고 잘하기 위해서

논문을 찾아봤습니다.

Dense Passage Retrieval for Open-Domain Question Answering

EMNLP 2020, facebook research

- 일반적인 Open Domain QA의 경우 BM25와 같은 희소 벡터 모델을 통해 문서 후보를 선택하고 이후 밀집 벡터 검색으로 passage 단위의 후보를 찾음
- 하지만 우리는 밀집 벡터 모델만으로 검색이 구현가능 하다는 것을 보임
- 단순하게 구성된 bi encoder 구조에서 튜닝된 Lucene-BM25 보다 9~19% 우수한 SOTA의 성능을 보였다.
- 충분히 적은 학습 데이터로도 잘 동작 하였다.

DRP (Dense Passage Retriever)

학습 데이터 1000개면 BM25 스코어를 상회 하더라

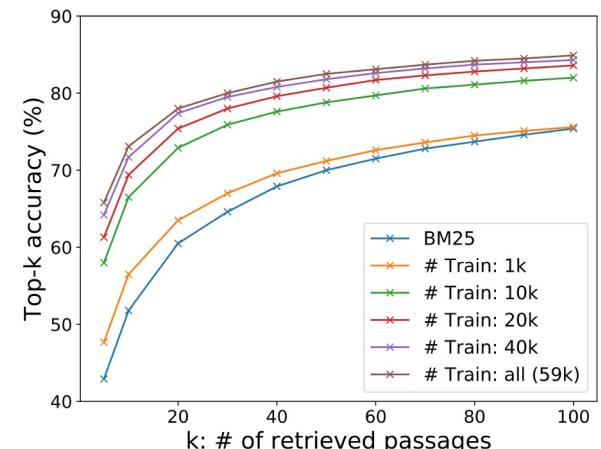
BM 25 결과를 negative 학습에 조금만 사용하니 성능이 크게 개선되었다.

구조도 단순하다.

in batch negative sampling 으로 최적화도 했다.

Training	Retriever	Top-20					Top-100				
		NQ	TriviaQA	WQ	TREC	SQuAD	NQ	TriviaQA	WQ	TREC	SQuAD
None	BM25	59.1	66.9	55.0	70.9	68.8	73.7	76.7	71.1	84.1	80.0
Single	DPR	78.4	79.4	73.2	79.8	63.2	85.4	85.0	81.4	89.1	77.2
	BM25 + DPR	76.6	79.8	71.0	85.2	71.5	83.8	84.5	80.5	92.7	81.3
Multi	DPR	79.4	78.8	75.0	89.1	51.6	86.0	84.7	82.9	93.9	67.6
	BM25 + DPR	78.0	79.9	74.7	88.5	66.2	83.9	84.4	82.3	94.1	78.6

Table 2: Top-20 & Top-100 retrieval accuracy on test sets, measured as the percentage of top 20/100 retrieved passages that contain the answer. *Single* and *Multi* denote that our Dense Passage Retriever (DPR) was trained using individual or combined training datasets (all the datasets excluding SQuAD). See text for more details.



하지만 - Dense Passage Retriever

Open QA 에 적합한 방식이 앱서비스 검색에 바로 적용 가능할지 알 수 없었음

QA 데이터 구축은 안하고 싶었음

단어 임베딩은 w2v glove fasttext 모두 별로 였음

그리고 BERT를 그냥 임베딩으로 쓰면 더 안좋았음

검색하는 초록 회사 자체 임베딩은 매우 품질이

좋았지만 이젠없음.

검색도 추천같은 coldstart model 이 필요하더라.



- Espresso? But I ordered a cappuccino!

- Don't worry, the cosine distance between them is so small that they are almost the same thing.

그래서 - Dense Passage Retriever

well define 된 모델을 활용해서 구조를 만드는것이 목표

SOTA(State-of-the-art) 모델은 고려하지 않음

단순함에 대한 사랑

2년 이상 지난 논문만 훑음

데이터 구축을 하지 않는다. 목표는 토스 데이터는 제로샷을, 백본과 대부분의 파인튜닝은 공개 학습셋만으로 커버하는것

NLI , STS, BM25 top positive 활용해서 multitask로 학습한다. 다만 bm25는 조금만 넣는다.

앞서서 - Dense Passage Retriever

대부분의 모델 성능은 bert-base 급 되는 110M급 LM에서 얻는다고 가정

DRP 논문에서 제시된 negative log likelihood loss 는 적당한 triplet loss로 대체한다.

학습 세트를 줄이기 위해서 현 구현에서는 classification 을 위한 softmax의 cross entropy를 사용

Triplet Objective Function. Given an anchor sentence a , a positive sentence p , and a negative sentence n , triplet loss tunes the network such that the distance between a and p is smaller than the distance between a and n . Mathematically, we minimize the following loss function:

$$\max(||s_a - s_p|| - ||s_a - s_n|| + \epsilon, 0)$$

with s_x the sentence embedding for $a/n/p$, $\|\cdot\|$ a distance metric and margin ϵ . Margin ϵ ensures that s_p is at least ϵ closer to s_a than s_n . As metric we use Euclidean distance and we set $\epsilon = 1$ in our experiments.

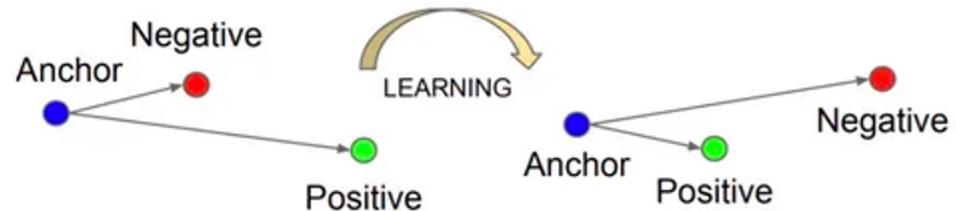
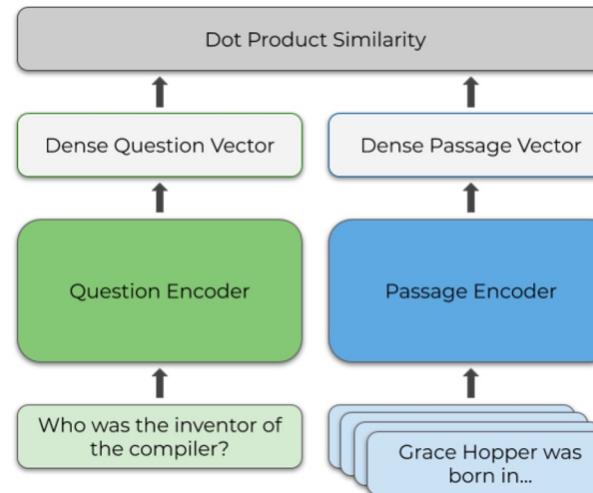


Figure 3. The **Triplet Loss** minimizes the distance between an *anchor* and a *positive*, both of which have the same identity, and maximizes the distance between the *anchor* and a *negative* of a different identity.

Architecture

DPR 원문 구조에서는 두가지 인코더를 모두 파인 튜닝함

이는 Open QA가 태스크가 충분히 많은 데이터를 가지고 있고 어려운 문제이기 때문
토스의 컨텐츠로는 규모로는 아직 어려운 접근



Architecture

DPR 자체의 인코더가 디코더가 잘 동작하는 것은 QA 데이터의 파인튜닝보단 triplet loss에 의한 metric learning 의 효과일 가능성이 높다고 가정

즉 이중 인코더의 파인 튜닝을 NLI (Natural language inference) 학습을 통한 triplet loss로 구성해본다. LM의 성능을 담보로한 단일 인코더 문제로 변환

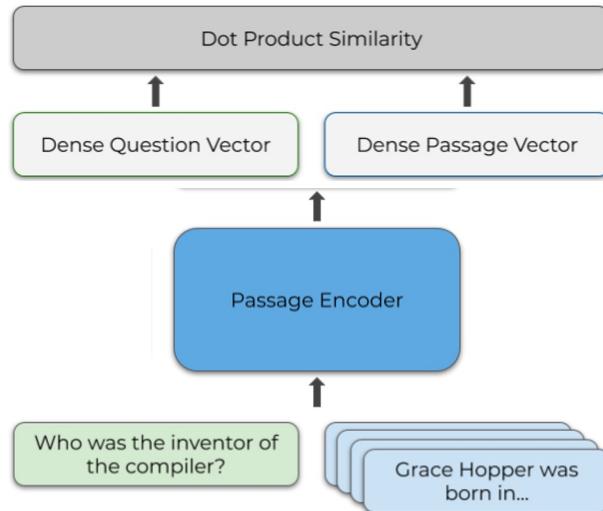


Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

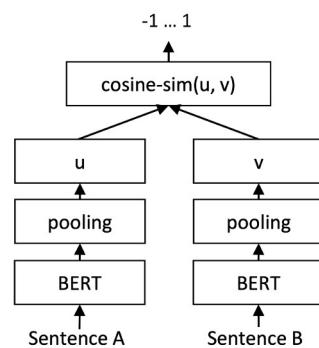
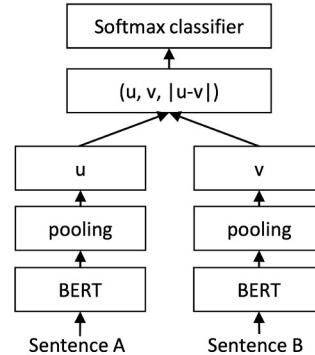


Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.

Architecture

카카오 브레인에서 공개된 NLI + STS 데이터 세트가 있었고
<https://github.com/kakaobrain/KorNLUDatasets>

해당 데이터를 통해 적당한 triplet loss 를 학습하면 DPR의 negative log-likelihood 를 대체 가능했음

Triplet Objective Function. Given an anchor sentence a , a positive sentence p , and a negative sentence n , triplet loss tunes the network such that the distance between a and p is smaller than the distance between a and n . Mathematically, we minimize the following loss function:

$$\max(\|s_a - s_p\| - \|s_a - s_n\| + \epsilon, 0)$$

with s_x the sentence embedding for $a/n/p$, $\|\cdot\|$ a distance metric and margin ϵ . Margin ϵ ensures that s_p is at least ϵ closer to s_a than s_n . As metric we use Euclidean distance and we set $\epsilon = 1$ in our experiments.

Let $\mathcal{D}^- = \{(q_i, p_i^+, p_{i,1}^-, \dots, p_{i,n}^-)\}_{i=1}^m$ be the training data that consists of m instances. Each instance contains one question q_i and one relevant (positive) passage p_i^+ , along with n irrelevant (negative) passages $p_{i,j}^-$. We optimize the loss function as the negative log likelihood of the positive passage:

$$\begin{aligned} & L(q_i, p_i^+, p_{i,1}^-, \dots, p_{i,n}^-) \\ &= -\log \frac{e^{\text{sim}(q_i, p_i^+)}}{e^{\text{sim}(q_i, p_i^+)} + \sum_{j=1}^n e^{\text{sim}(q_i, p_{i,j}^-)}}. \end{aligned} \tag{2}$$

Positive and negative passages For retrieval problems, it is often the case that positive examples are available explicitly, while negative examples need to be selected from an extremely large pool.

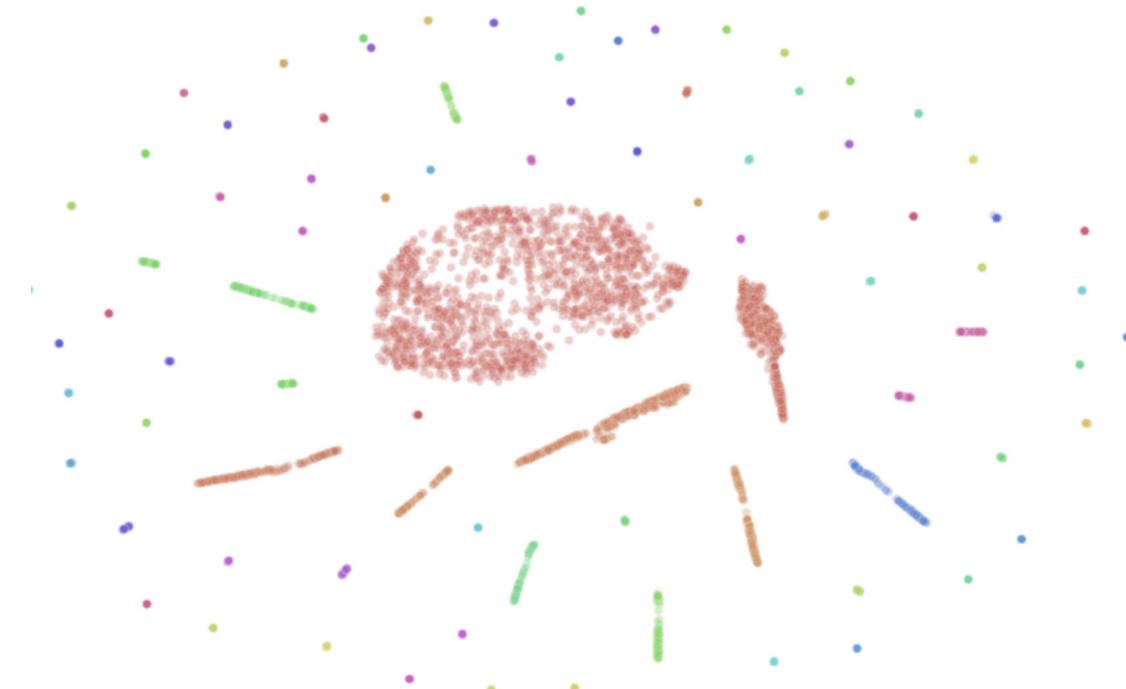
Architecture

다음과 같은 문장 임베딩이 있을때



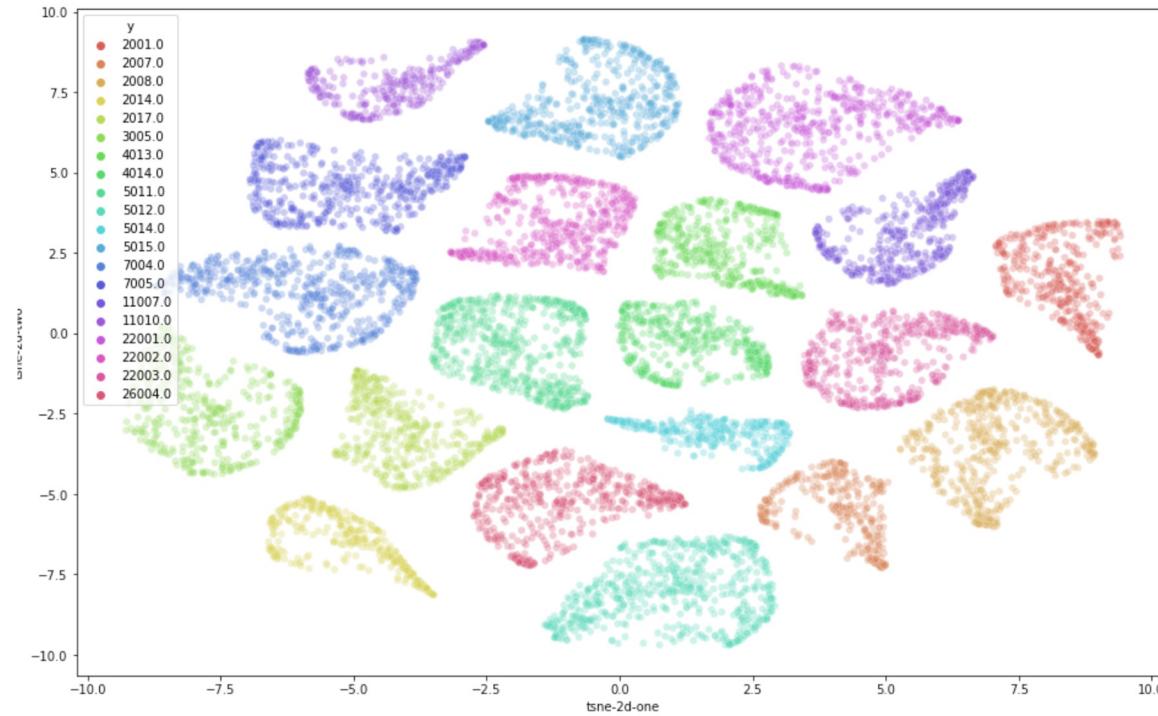
Architecture

metric learning 이 적용되지 않은 일반적인 classification 의 결과 manifold



Architecture

metric learning 이 적용된 manifold

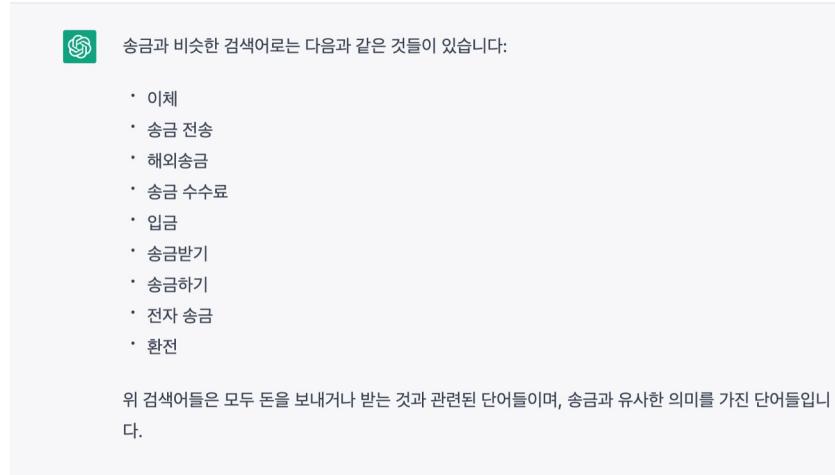


Architecture

기존의 검색 문제를 Query -> Query' 문제로 가볍게 정의

범용적인 Sentence Embedding 이 제공하는 zeroshot approach 을 최대한 이용

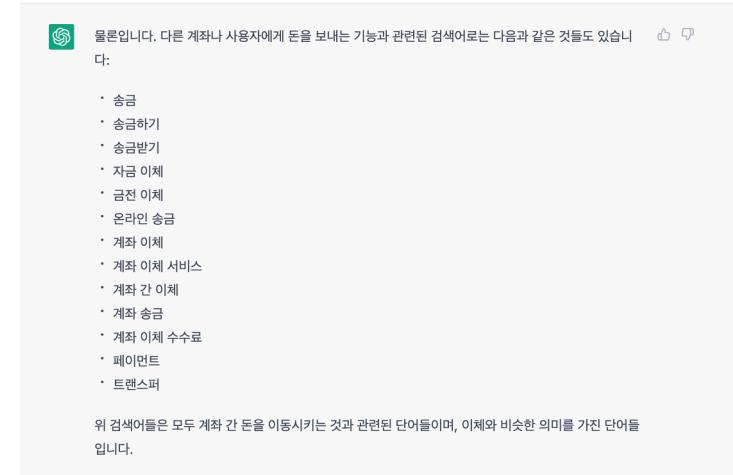
Query' 의 생성은 생성 언어 모델을 사용하여도 충분



송금과 비슷한 검색어로는 다음과 같은 것들이 있습니다:

- 이체
- 송금 전송
- 해외송금
- 송금 수수료
- 입금
- 송금받기
- 송금하기
- 전자 송금
- 환전

위 검색어들은 모두 돈을 보내거나 받는 것과 관련된 단어들이며, 송금과 유사한 의미를 가진 단어들입니다.



송금입니다. 다른 계좌나 사용자에게 돈을 보내는 기능과 관련된 검색어로는 다음과 같은 것들도 있습니다:

- 송금
- 송금하기
- 송금받기
- 자금 이체
- 금전 이체
- 온라인 송금
- 계좌 이체
- 계좌 이체 서비스
- 계좌 간 이체
- 계좌 송금
- 계좌 이체 수수료
- 페이먼트
- 트랜스퍼

위 검색어들은 모두 계좌 간 돈을 이동시키는 것과 관련된 단어들이며, 이체와 비슷한 의미를 가진 단어들입니다.

Architecture

사내 제품과 서비스에 자연어 힌트와 서술을 손쉽게 배치하도록 지원

search view

← 이전으로 돌아가기

송금 (379)

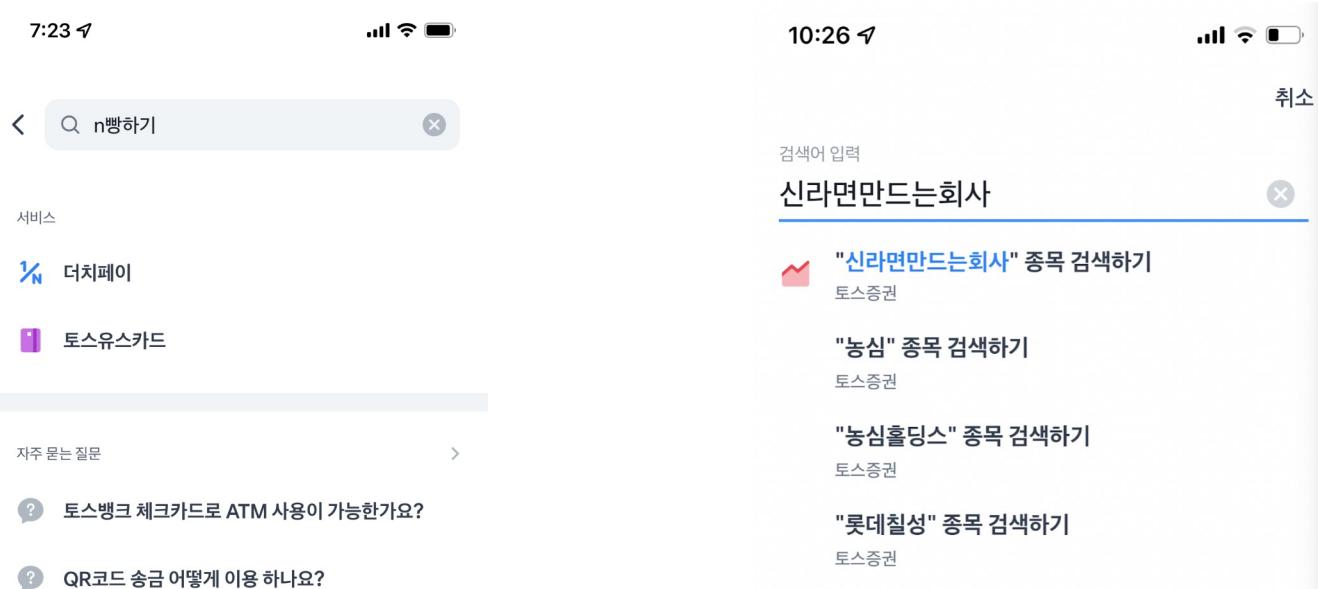
기본 정보

수정하기

ID	379
PID	0
서비스 ID	15
이름	송금
전체 경로	송금
서비스 오픈	true
학습 데이터	<input type="radio"/> 다른 계좌나 사용자에게 돈을 보내는 기능 <input type="radio"/> 이제 <input type="radio"/> 계좌 송금 <input type="radio"/> 계좌 이제 <input type="radio"/> 돈보내기
메모	
서비스 노출 영역	RECO
Threshold Override 값	0.71

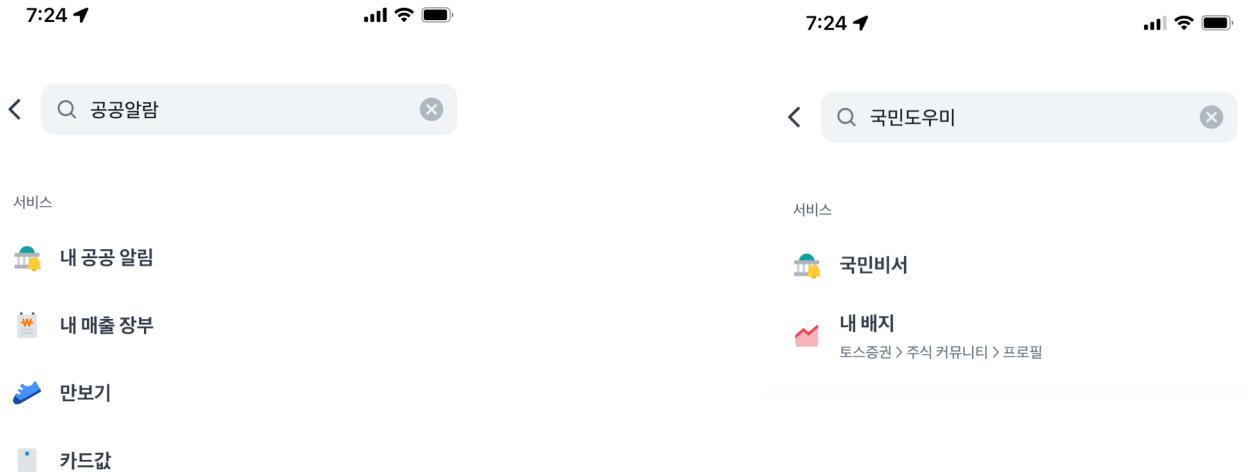
Architecture - Cherry Picking 딥러닝이 제일 쉬웠어요

1000개 미만의 문장과 단어로 운영 가능한 탁월한 Semantic Retrieval 성능



Architecture - Cherry Picking 딥러닝이 제일 쉬웠어요

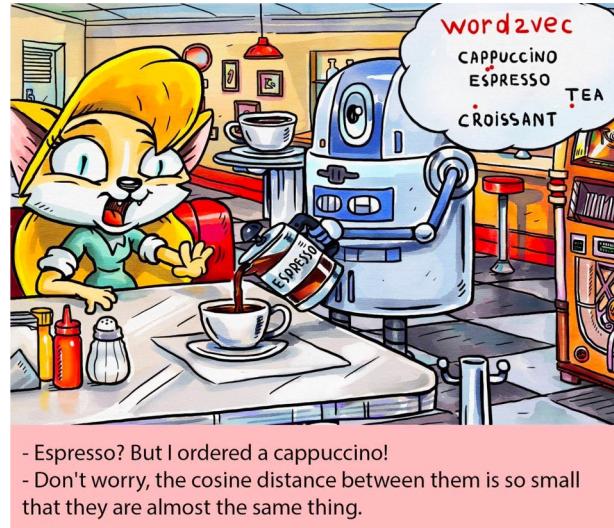
유저의 의도에 가장 가까운 명칭으로 서비스를 적절하게 노출



Architecture - 한계점

Sparse Embedding에서 기대하는 Lexical 성능의 저조

DPR의 구현과 같은 bm25 결과를 일부 활용하거나 적절한 거리 함수 기반 보정이 필요



Architecture - Engineering 적인 문제점

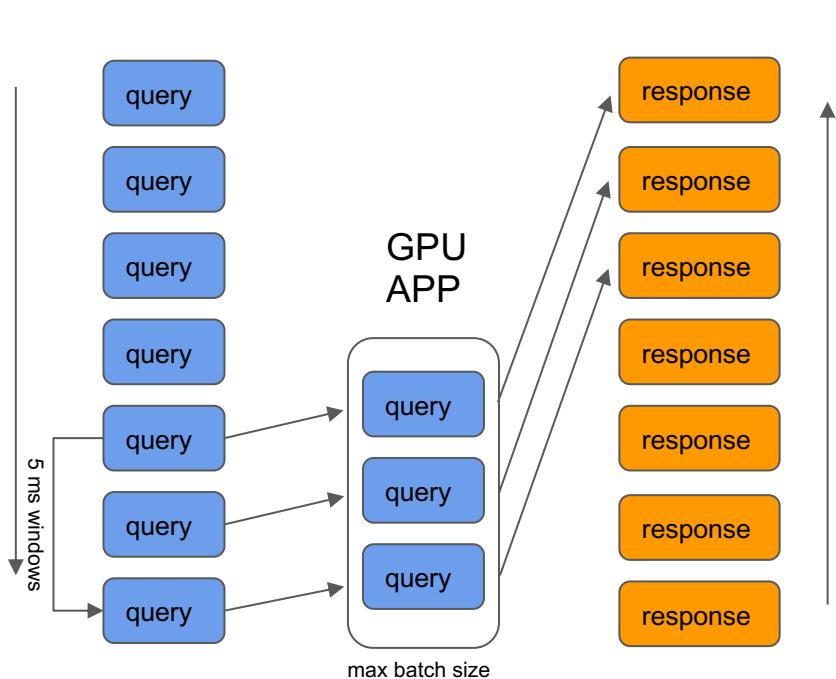
- BERT 110M 수준의 모델은 연산 비용이 높음
 - GPU 를 사용한 단순한 구조의 real time inference로 약 50~100 QPS (query per second) 성능
- ANN등 벡터 유사도 연산에 다수의 서빙과 인프라가 필요
 - 주기적으로 indexing 처리와 별도 서빙 서버등 필요
 - 관련된 인프라가 많고 복잡할 경우 MSA의 저주에 걸릴 수 있다.
- ANN의 활용 효율이 많이 떨어짐
 - ANN은 효율적인 벡터의 숫자가 존재 약 적어도 10만 이상의 타겟 벡터에서 의미가 있음
 - 효과적으로 근사되는 차원의 범위가 존재 예를 들면 20~ 256 정도가 일반적
 - 768 ~ 2048 혹은 30000 차원 수준의 dense vector 검색에서는 recall 성능 감소가 아쉬움

Architecture - Engineering 적인 해결 방법

- 성능에 가장 중요한 Dynamic batch는 간단히 jvm 수준에서 처리
 - Reactive Streams을 이용해서 dynamic batch 를 WAS 수준으로 간단히 구현
 - pod당 5배 이상의 QPS 성능 증가
- 벡터 검색 엔진과 인덱싱등 인프라도 java 코드 로직으로 간단히 처리
 - Vector 유사도 계산을 embedding extraction과 동시에 dynamic batch 세션에서 그대로 처리
 - 단일 GPU (RTX 8000, A100) 1 pod 당 500 ~ 1500 TPS 이상의 응답 성능 확보
 - 단일 was에서 단위에서 gpu, cpu 모두 scale down, up 손쉬운 구성 확보
- Gpu를 사용한다면 고차원, 적은 숫자의 벡터 유사도는 직접 계산해도 OK
 - 서비스 내부 검색은 5 ~ 10 만 정도 수준의 대상 vector를 가지고 있으므로 gpu의 in-memory 에서 최대한 고정하고 처리
 - 사전 정규화 후 batch dot 을 처리한후 상위 벡터만 tensor 단위로 jvm으로 리턴

Architecture - Dynamic batching

동시성 처리와 stream processing에 적합한 jvm에서 queuing과 batch를 구현



benchmark

CPU : Ryzen 9 3950X
GPU : Nvidia Geforce rtx 3090 (495.29.05)
ENV : docker.io/nvidia/cuda:11.3.1-cudnn8-runtime-ubuntu18.04
CMD : hey -n 10000 -c 256 http://servicehost:8080/bert?query=이 영화는 최고의 영화입니다

koelectra-small
<https://huggingface.co/daekeun-ml/koelectra-small-v3-nsmc>

Sequence length	8	16	32	64	128	256	512
Requests/sec	19035	14434	10400	6792	4132	1912	788
latency avg (ms)	13	17	23.6	37.2	61.5	71.1	161.4

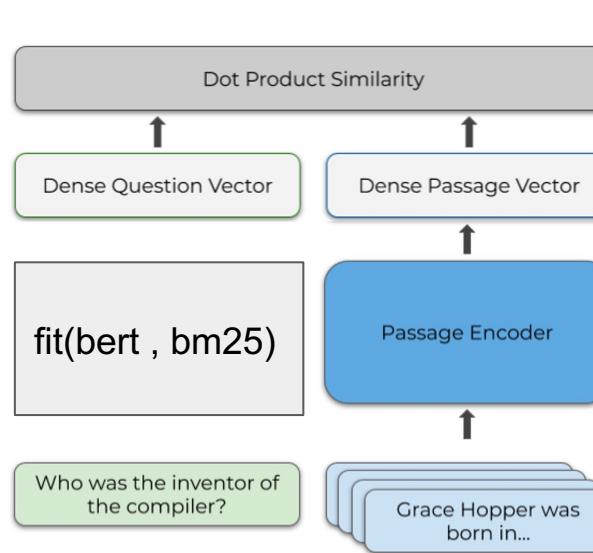
bert-base
<https://huggingface.co/sangrimlee/bert-base-multilingual-cased-nsmc>

Sequence length	8	16	32	64	128	256	512
Requests/sec	8608	5595	3314	1799	904	429	187
latency avg (ms)	27	44.6	76.1	144.4	280.8	587.1	1340.3

그리고 통합 검색으로

Query -> Query' 를 찾는 문제는 충분히 robust 하며 적은 데이터로 동작 가능

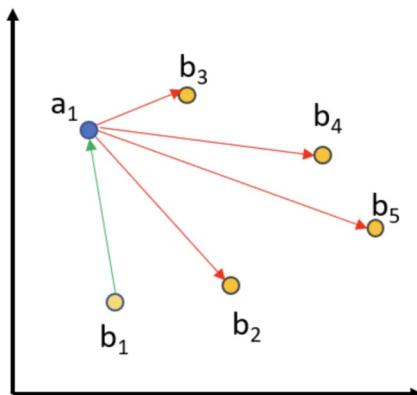
하지만 결국은 Query -> Documents 를 찾는 QA 문제를 잘 풀어야 Contents 검색이 가능



그리고 통합 검색으로

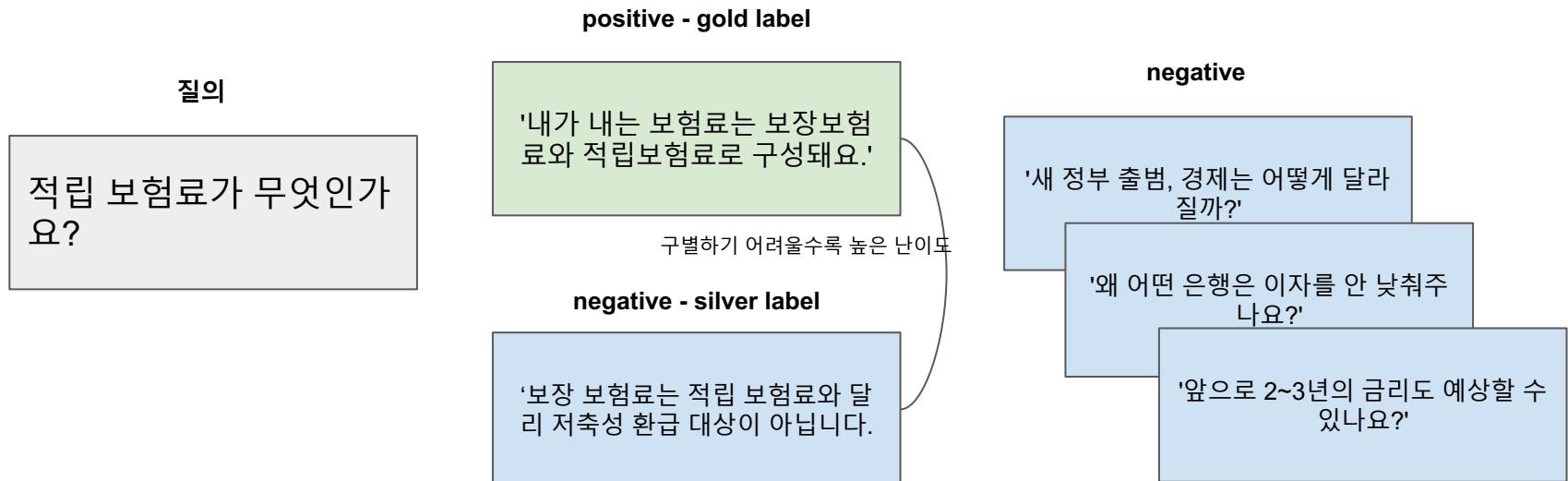
LM을 MultipleNegativesRankingLoss 를 통해 bm25 결과로 초기 세팅하고
사내 테스트와 AB 테스트로 지속적으로 implicit feedback 을 수집하는 추천 태스크
간주

Efficient Natural Language Response Suggestion for Smart Reply <https://arxiv.org/pdf/1705.00659.pdf>



그리고 통합 검색으로

추가로 LLM을 통한 데이터 augmentation, BM25를 통한 negative sample 생성
gold - silver 의 차이를 최소화하는 hard negative를 얼마나 잘 세팅하는지가 중요



그리고 통합 검색으로

데이터의 실시간 갱신과 인덱싱을 위해 Vector Search Engine을 동시활용
SLA의 요구 사항에 따라 가변적으로 사용

그리고 통합 검색으로

내부 임직원에게 큰 사랑을 받으며

운영 로그를 통해 지속적으로 개선, 재학습 되고

자소 단위까지 타이핑 즉시 검색되며

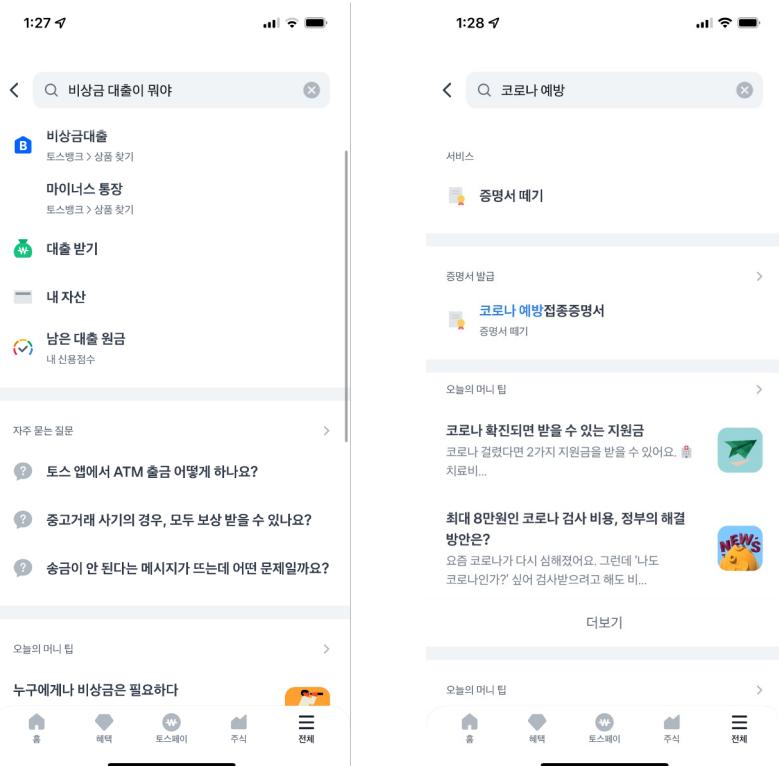
오 입력과 유사 유의어에 강인하고

인프라에 유연한 배포 환경을 가진 통합 검색을

비교적 즐겁고 빠르게 만들 수 있었습니다.

special thank you

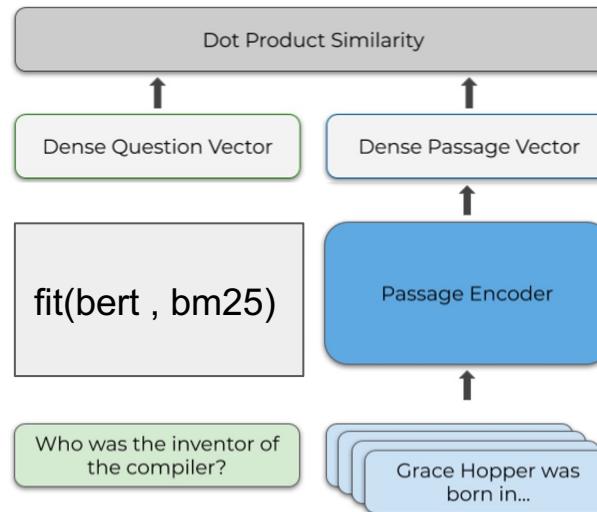
Toss - Modeling Team 오연택 님



앞으로 생각하는 것들

검색은 결국 자연어를 통한 Suggest의 영역으로 가야하며

Cold Start - Recommender System의 좋은 접근 방법 중 하나이고



앞으로 생각하는 것들

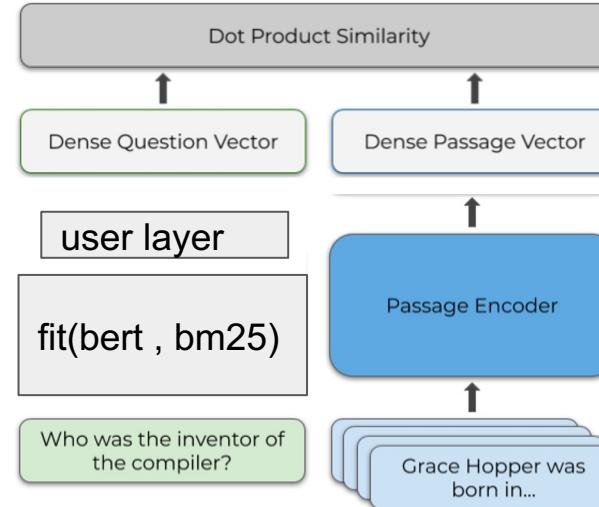
NN의 구조적 유연성을 통해서 검색과 추천의 통합이 용이하고

on-device 수준에서의 Federated Learning 형태의 개인화가 가능하고

대규모 개인화의 시스템 비용이 현저히 감소할 것

예를 들어

6kb 수준의 간단한 개인화 nn layer를
디바이스별로 배치한다면 혹은
디바이스에서 서버를 통해 gradient만
수신해서 직접 학습한다면 ..



Q & A