

Dense Passage Retrieval for Open-Domain Question Answering

EMNLP 2020

Karpukhin, Vladimir, et al

2022.01.27

지능융합학과 정민지

Contents

1. Summary
2. Information about ODQA
3. Passage Retrieval
4. Dense Embedding
5. Training Dense Encoder
6. End-to-end QA System

1. Summary

1. 2개의 BERT encoder로 이루어진 dense retrieval 방법 제안
2. Pretraining 없이 기존의 Question-Answering쌍을 fine-tuning하는 것 만으로도 성능이 잘 나옴
3. DPR 사용시 retriever와 end-to-end QA system 모두 성능이 올라감을 확인함

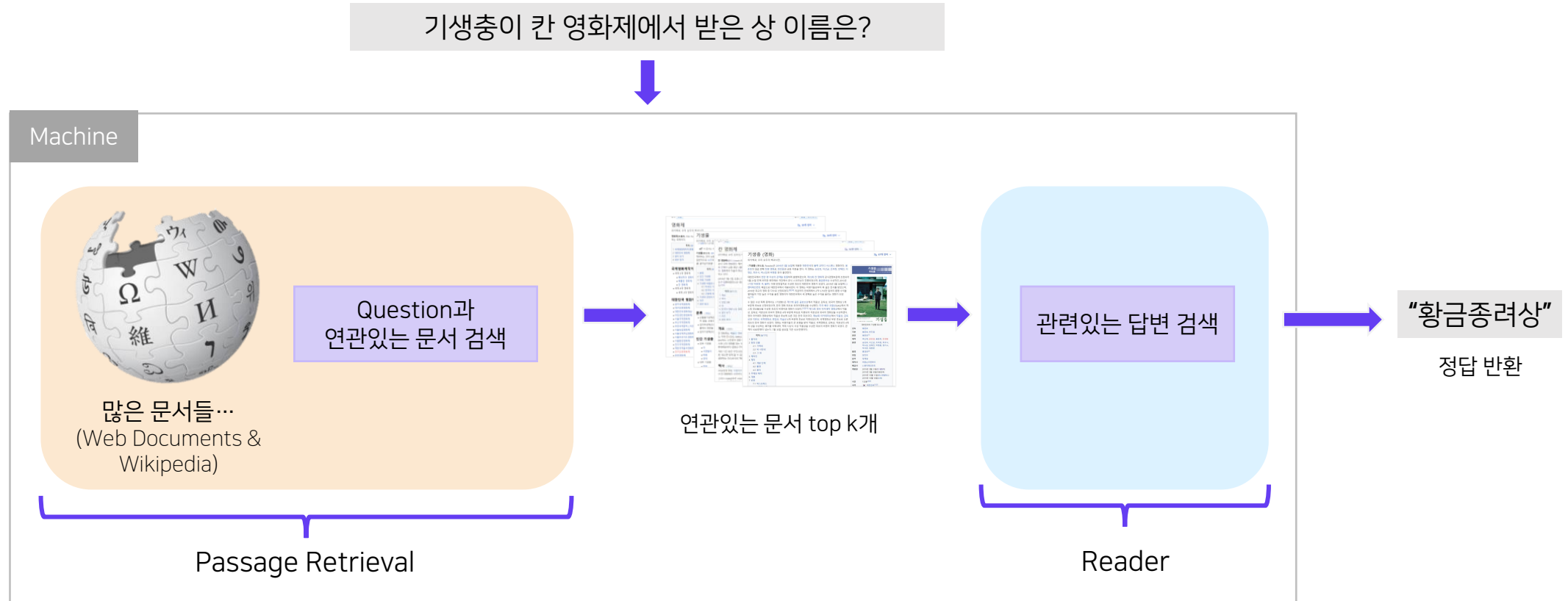
2. Information about ODQA

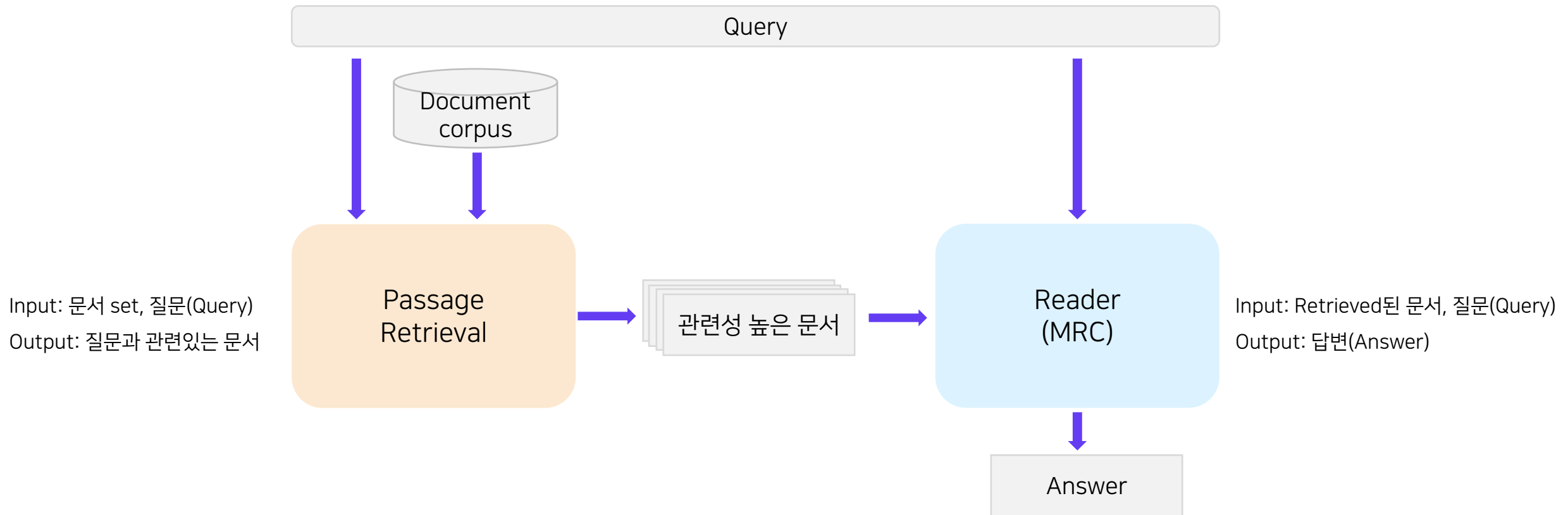
Extraction-based MRC

2. Information about ODQA

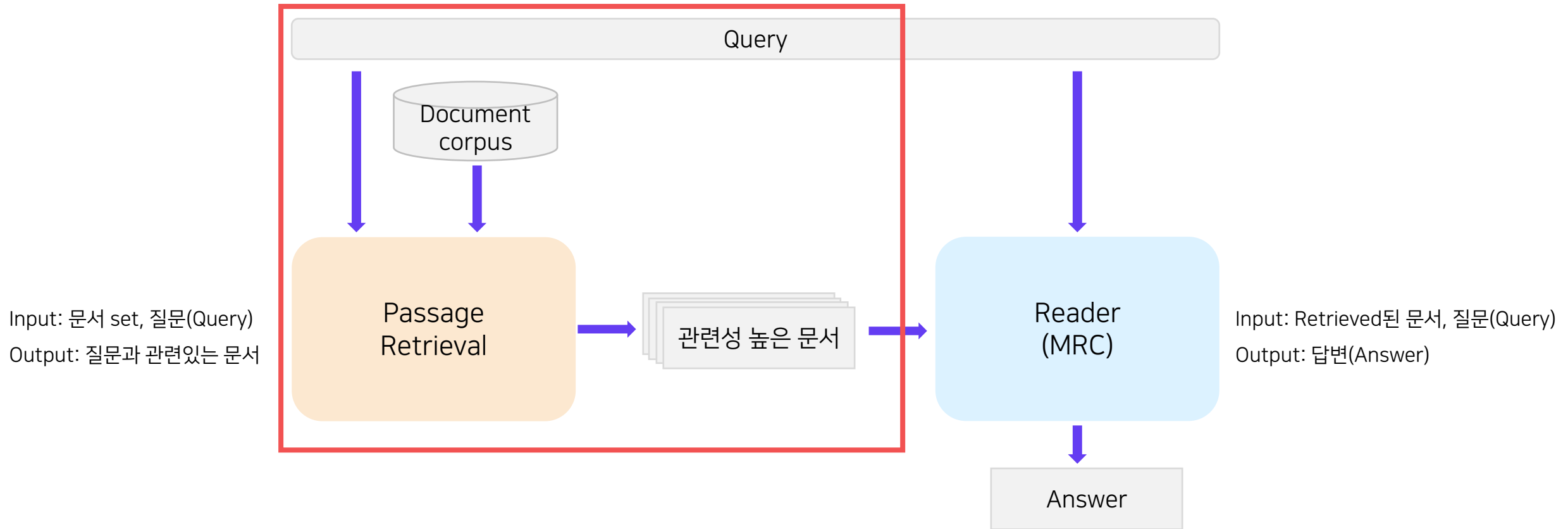
Query (Question)	기생충이 칸 영화제에서 받은 상 이름은?
Context	《기생충》(Parasite)은 2019년 5월 30일에 개봉한 대한민국의 블랙 코미디 서스펜스 영화이다. 봉준호의 일곱 번째 장편 영화로, ... 칸 영화제에서 황금종려상 을 수상하였다. 2013년 ...

- Context에서 질문에 대한 답을 찾는 Task
- 질문의 답변이 **주어진 지문 내에 하나의 연속된 span**으로 존재





- **Retriever**과 **Reader** 두 단계로 구성됨
- **Retriever**: DB의 많은 문서 중 Question과 연관있는 **Top-k개 문서**를 찾음(search)
- **Reader**: 검색된 Top-k 문서에서 **Question에 대한 답**을 찾아냄

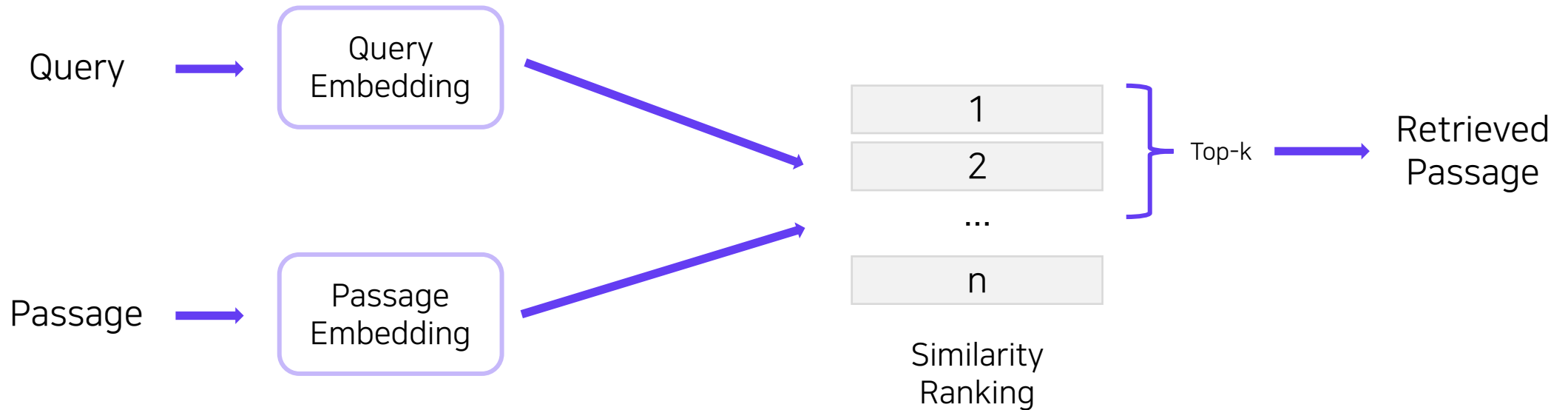


- **Retriever**과 **Reader** 두 단계로 구성됨
- **Retriever**: DB의 많은 문서 중 Question과 연관있는 **Top-k개 문서**를 찾음(search)
- **Reader**: 검색된 Top-k 문서에서 **Question에 대한 답**을 찾아냄

3. Passage Retrieval

Overview of Passage Retrieval

3. Passage Retrieval



Query와 Passage를 Embedding한 뒤 유사도 ranking을 매기고, 유사도가 가장 높은 (k개의) 문서를 반환함

Passage Embedding

3. Passage Retrieval

- Passage를 벡터로 변환하는 것

Passage

기생충 (영화)

위키백과, 우리 모두의 백과사전.

《기생충》(韓語: Parasite)은 2019년 5월 30일에 개봉한 대한민국의 블랙 코미디 서스펜스 영화이다. 봉준호의 일곱 번째 장편 영화로, 안진원과 공동 각본을 썼다. 이 영화는 송강호, 이선균, 조여정, 장혜진, 이정은, 최우식, 박소담과 박영준 등이 출연한다.

대한민국에서 한만 명 이상의 관객을 동원하여 흥행하였으며, 제72회 칸 영화제 공식경쟁부분에 초청되어 5월 21일 칸에 위치한 휘미에르 극장에서 공식 스크리닝이 진행되었으며, 황금종려상 수상작인 2013년 《가장 아름다운 세 불꽃》 이래 만장일치로 수상한 최초의 대한민국 영화가 되었다. 2019년 5월 30일에 CJ 엔터테인먼트 배급으로 대한민국에서 개봉되었다. 이 영화는 비평가들로부터 폭 넓은 찬사를 받았으며, 2019년 최고의 영화 중 다수로 선정되었다.^{[9][10]} 지금까지 전세계에서 2억 5,755만 달러의 흥행 수익을 벌여들어 가장 높은 수익을 올린 영화이자 대한민국에서 세 번째로 높은 수익을 올리는 영화가 되었다.^[11]

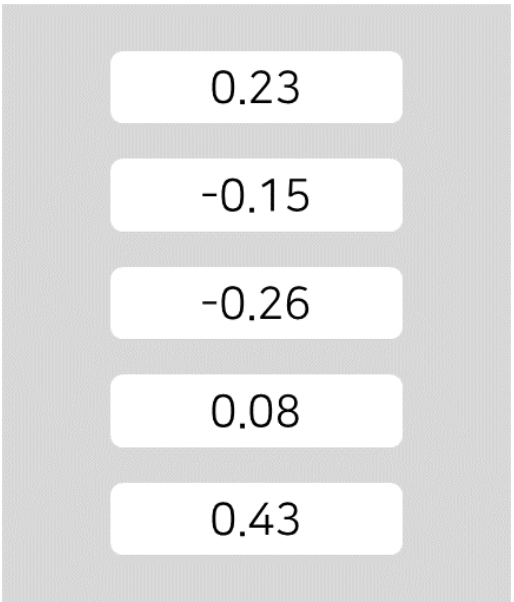
수 많은 수상 목록 중에서도 《기생충》은 제77회 골든 글로브상에서 작품상, 감독상, 외국어 영화상 3개 부문에 후보로 선정되었으며, 한국 영화 최초로 외국어영화상을 수상했다. 미국 배우 조합상(SAG)에서 캐스팅 앙상블상을 수상한 최초의 비영어권 영화가 되었다.^{[12][13]} 제71회 영국 아카데미 영화상에서 작품상, 감독상, 각본상과 외국어 영화상 4개 부문에 후보로 지명되어 각본상과 외국어 영화상을 수상하였다. 영국 아카데미 영화상에서 작품상 후보에 오른 것은 한국 최초이다. 제92회 아카데미상에서 작품상, 감독상과 각본상, 국제영화상, 편집상, 미술상 6개 부문에 후보로 지명되었으며, 국제영화상 부문 후보로 오른 최초의 한국 영화가 되었다. 영화는 비평가들의 큰 호평을 받아 작품상, 국제영화상, 감독상, 각본상의 4개의 상을 수상하는 쾌거를 이뤄내며, 역대 시상식 사상 작품상을 수상한 최초의 비영어 영화가 되었다. 관객이 1000만명이 넘는다. 7월 30일 기준으로 1031만명이다.

목차 (숨기기)

- 1 줄거리
- 2 등장 인물
 - 2.1 가족들
 - 2.2 박 사장네
 - 2.3 그 외
- 3 제작진
- 4 제작
 - 4.1 개발 단계
 - 4.2 촬영
 - 4.3 음악
- 5 주제와 해석
- 6 개봉
- 7 반응
 - 7.1 박스오피스



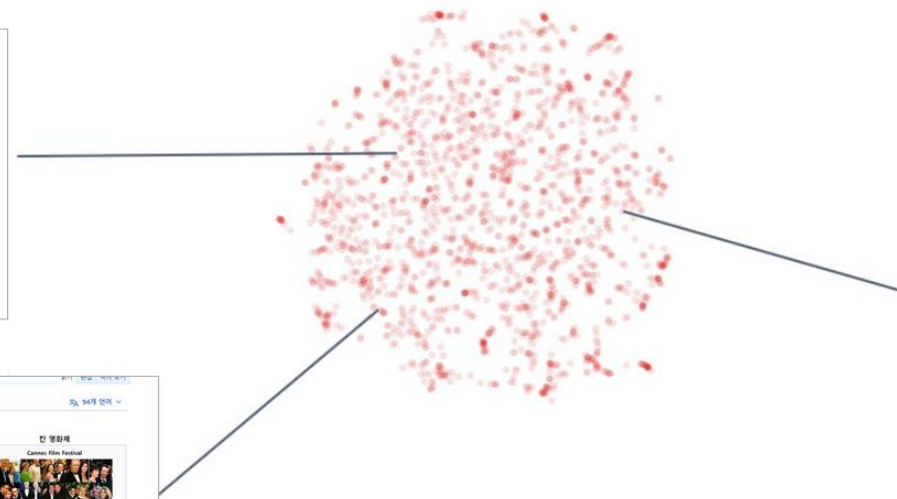
Passage Embedding



Passage Embedding Space

3. Passage Retrieval

- Passage Embedding의 벡터 공간 (지문을 숫자로 mapping → 고차원의 벡터 공간)
- 벡터화된 Passage를 이용하여 Passage간 유사도 등을 알고리즘으로 계산할 수 있다.



Sparse Embedding

TF-IDF (Term Frequency – Inverse Document Frequency)

문서 제목	문서 내용
음식	나는 떡볶이를 좋아한다
계절	나는 가을을 가장 좋아한다
스포츠	나는 야구 관람을 가장 좋아한다

	나는	떡볶이를	좋아한다	가을을	가장	야구	관람을
음식	1.0	1.0	1.0	0.0	0.0	0.0	0.0
계절	1.0	0.0	1.0	1.0	1.0	0.0	0.0
스포츠	1.0	0.0	1.0	0.0	1.0	1.0	1.0

TF(t,d): 단어의 등장 빈도

	나는	떡볶이를	좋아한다	가을을	가장	야구	관람을
음식	0.0	1.1	0.0	1.1	0.4	1.1	1.1
계절	0.0	1.1	0.0	1.1	0.4	1.1	1.1
스포츠	0.0	1.1	0.0	1.1	0.4	1.1	1.1

IDF: 단어가 얼마나 중요한 정보를 담고있는가?

$$IDF(t) = \log \frac{N}{DF(t)}$$

N: 전체 문서의 개수
DF: t가 등장한 문서의 개수

	나는	떡볶이를	좋아한다	가을을	가장	야구	관람을
음식	0.0	1.1	0.0	0.0	0.0	0.0	0.0
계절	0.0	0.0	0.0	1.1	0.4	0.0	0.0
스포츠	0.0	0.0	0.0	0.0	0.4	1.1	1.1

$$TF(t,d) \times IDF(t)$$

각 문서가 가진 고유한 단어가 TF-IDF 값이 높은편

Sparse Embedding

3. Passage Retrieval

TF-IDF (Term Frequency – Inverse Document Frequency)

문서 제목	문서 내용
음식	나는 떡볶이를 좋아한다
계절	나는 가을을 가장 좋아한다
스포츠	나는 야구 관람을 가장 좋아한다

	나는	떡볶이를	좋아한다	가을을	가장	야구	관람을
음식	1.0	1.0	1.0	0.0	0.0	0.0	0.0
계절	1.0	0.0	1.0	1.0	1.0	0.0	0.0
스포츠	1.0	0.0	1.0	0.0	1.0	1.0	1.0

TF(t,d): 단어의 등장 빈도

	나는	떡볶이를	좋아한다	가을을	가장	야구	관람을
음식	0.0	1.1	0.0	1.1	0.4	1.1	1.1
계절	0.0	1.1	0.0	1.1	0.4	1.1	1.1
스포츠	0.0	1.1	0.0	1.1	0.4	1.1	1.1

IDF: 단어가 얼마나 중요한 정보를 담고있는가?

$$IDF(t) = \log \frac{N}{DF(t)}$$

N: 전체 문서의 개수
DF: t가 등장한 문서의 개수

	나는	떡볶이를	좋아한다	가을을	가장	야구	관람을
음식	0.0	1.1	0.0	0.0	0.0	0.0	0.0
계절	0.0	0.0	0.0	1.1	0.4	0.0	0.0
스포츠	0.0	0.0	0.0	0.0	0.4	1.1	1.1

$$TF(t,d) \times IDF(t)$$

각 문서가 가진 고유한 단어가 TF-IDF 값이 높은편


* BM25

TF-IDF에서 문서의 길이를 고려한 방법으로
문서가 짧을 수록 해당 단어의 등장에 대한
희귀성을 인정해주는 방법

Sparse Embedding

3. Passage Retrieval

문서가 많아지면 0의 비율이 더 많아질 것! → 차원 수 더 증가



	나는	떡볶이를	좋아한다	가을을	가장	야구	관람을
음식	0.0	1.1	0.0	0.0	0.0	0.0	0.0
계절	0.0	0.0	0.0	1.1	0.4	0.0	0.0
스포츠	0.0	0.0	0.0	0.0	0.4	1.1	1.1

- BoW를 사용하기 때문에 특정 단어에 해당하는 부분만 non-zero가 된다.
 - 차원의 수가 매우 크다
 - Compressed format(non-zero인 부분의 위치, 값만 저장)으로 극복 가능
- 단어의 의미적 유사성을 고려하지 못한다.

4. Dense Embedding

Dense Embedding

4. Dense Embedding

	나는	떡볶이를	좋아한다	가을을	가장	야구	관람을
음식	0.0	1.1	0.0	0.0	0.0	0.0	0.0
계절	0.0	0.0	0.0	1.1	0.4	0.0	0.0
스포츠	0.0	0.0	0.0	0.0	0.4	1.1	1.1



	dim1	dim2	dim3
음식	1.10	6.42	0.23
계절	5.64	0.29	7.62
스포츠	8.43	0.01	1.93

- 더 작은 차원의 고밀도 벡터
- 각 차원이 특정 term에 대응되지 않음
- 대부분의 요소가 non-zero값

Sparse vs Dense

4. Dense Embedding



sparse

“How many provinces did the Ottoman empire contain in the 17th century?”

“What part of the atom did Chadwick discover?”



dense

“Who is the bad guy in lord of the rings?”

Sala Daker is an actor and stuntman from New Zealand. He is best known for portraying the villain Sauron in the Lord of the Rings trilogy by Peter Jackson.

Sparse

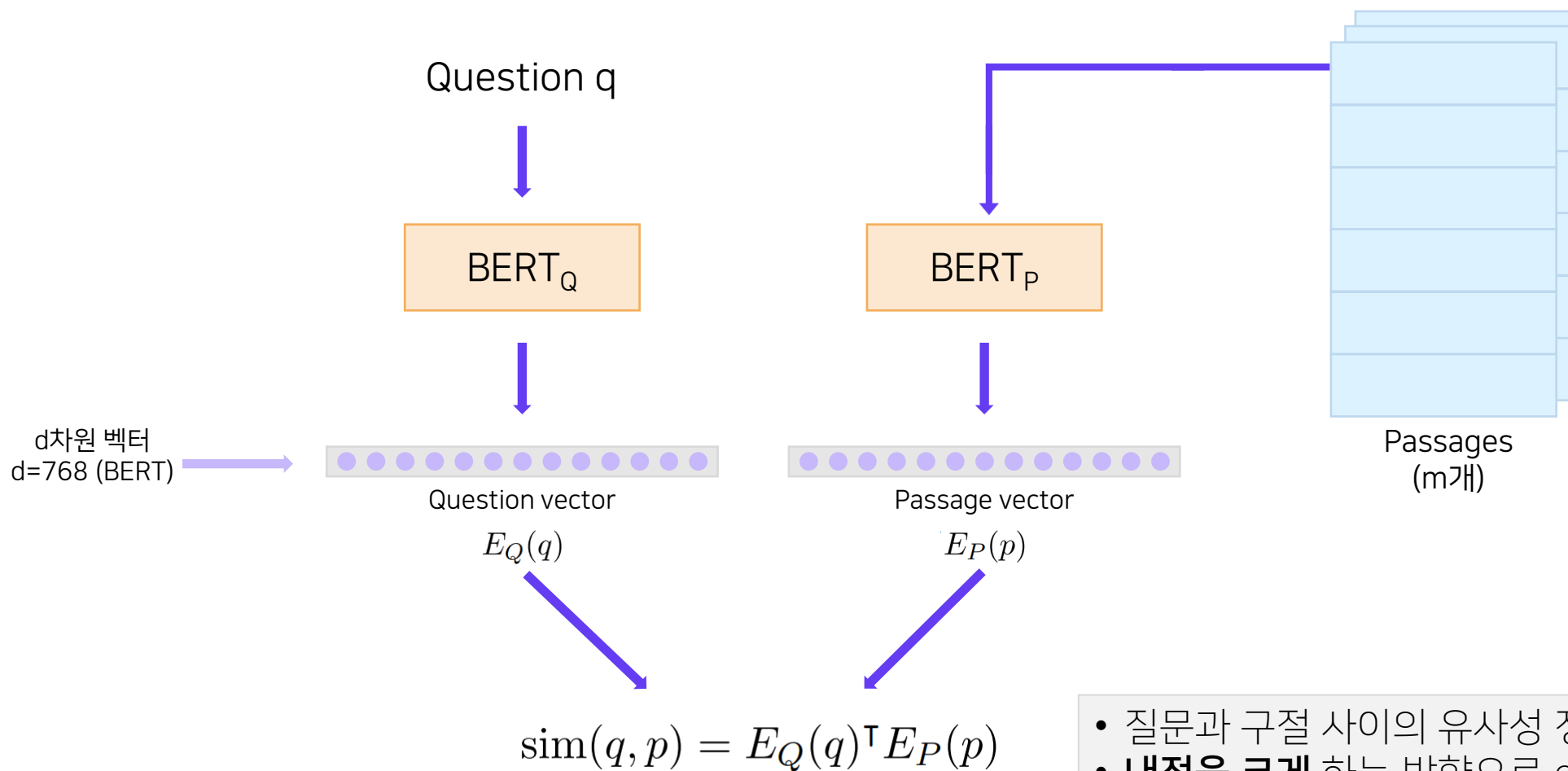
- Keyword가 정확하게 일치하는 경우 효율적임
- Question과 Context를 고차원의 vector 공간에서 나타냄

Dense

- 동의어: 완전히 다른 토큰으로 구성되더라도 서로 가까운 vector로 mapping됨 (단어의 유사성, 맥락 파악)
- 학습을 통해 Embedding을 만들며 기능을 조절하여 task-specific하게 학습 가능

Overview

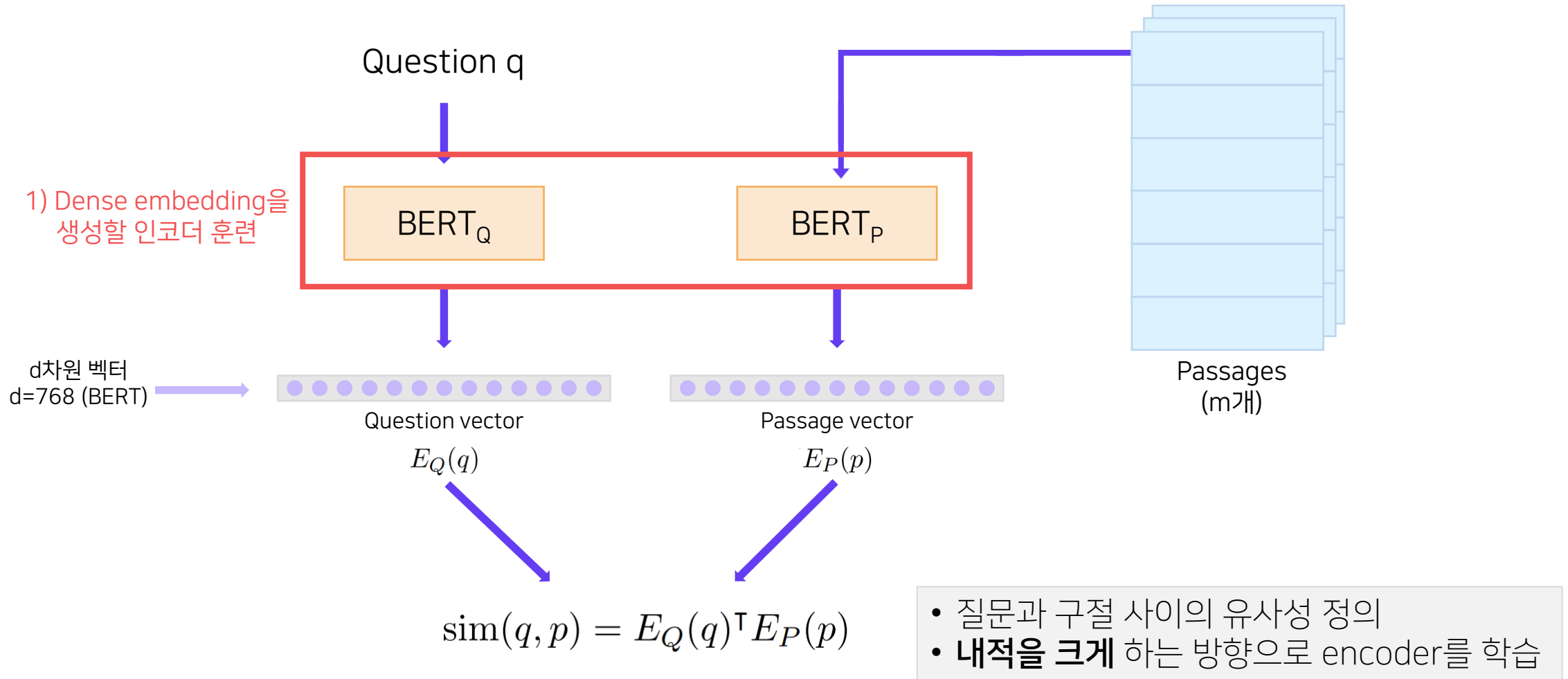
4. Dense Embedding



- 질문과 구절 사이의 유사성 정의
- **내적을 크게** 하는 방향으로 encoder를 학습

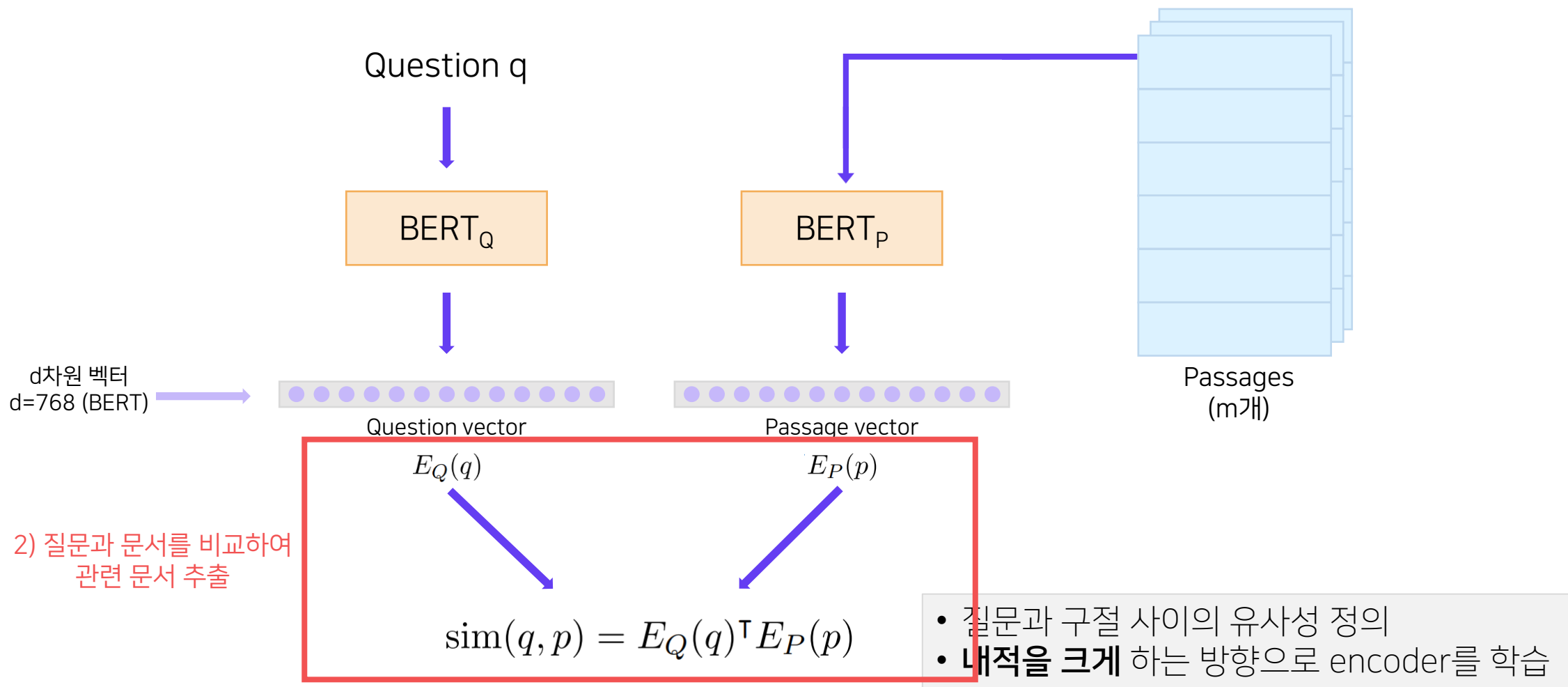
Overview

4. Dense Embedding



Overview

4. Dense Embedding



5. Training Dense Encoder

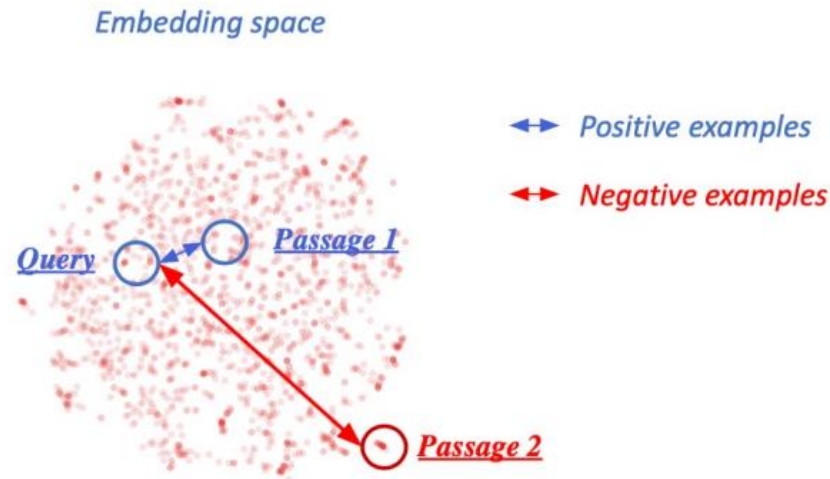
- 목표: 연관된 Question과 passage dense embedding간의 **inner product**를 높이는 것.
즉 **높은 유사도**를 갖게끔 하는 것
- 이 때 QA set만을 학습에 사용하기 위하여 기존 MRC 데이터셋을 학습하여 학습



Positive and Negative Sampling

- Negative sampling 이란?

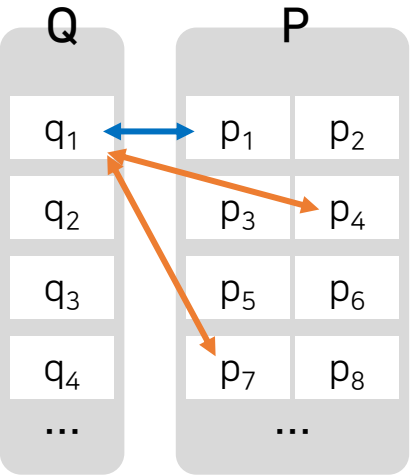
- 1) 연관된 question과 passage 간의 dense embedding 거리를 좁히는 것 (higher similarity) \Rightarrow **Positive**
- 2) 연관 되지 않은 question과 passage간의 embedding 거리는 멀어야 함 \Rightarrow **Negative**



Positive and Negative Sampling

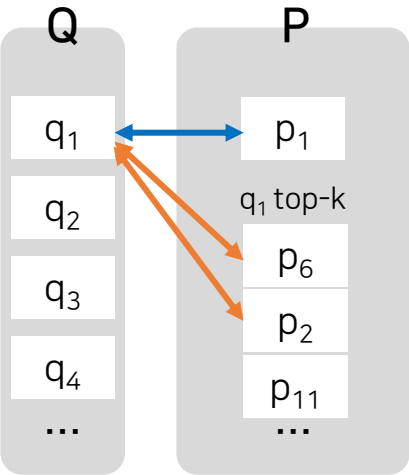
- How to select negative examples?

↔ Positive examples
↔ Negative examples



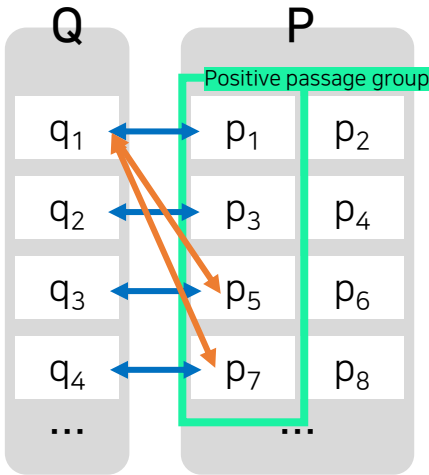
1) 랜덤 선택

Positive passage를 제외한 선택지 중
랜덤으로 선택한다.



2) BM25 기준으로 top-k 사용

정답을 포함하지 않는 passage중
BM25가 높은 것을 선택한다.



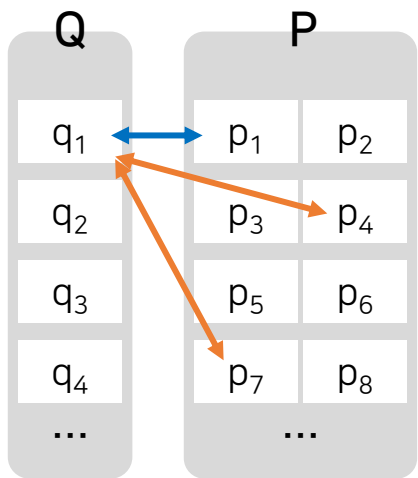
3) Gold 사용

다른 question의 positive passage에 해당하
는 것들 중 랜덤으로 선택한다.

Positive and Negative Sampling

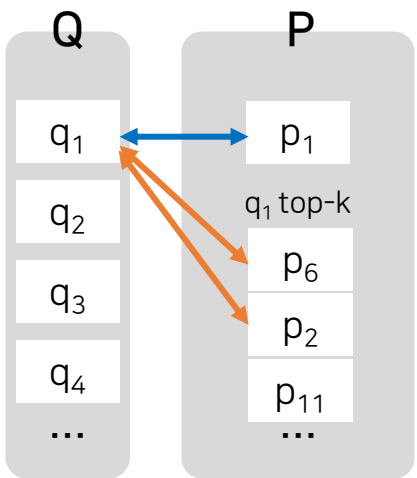
- How to select negative examples?

↔ Positive examples
↔ Negative examples



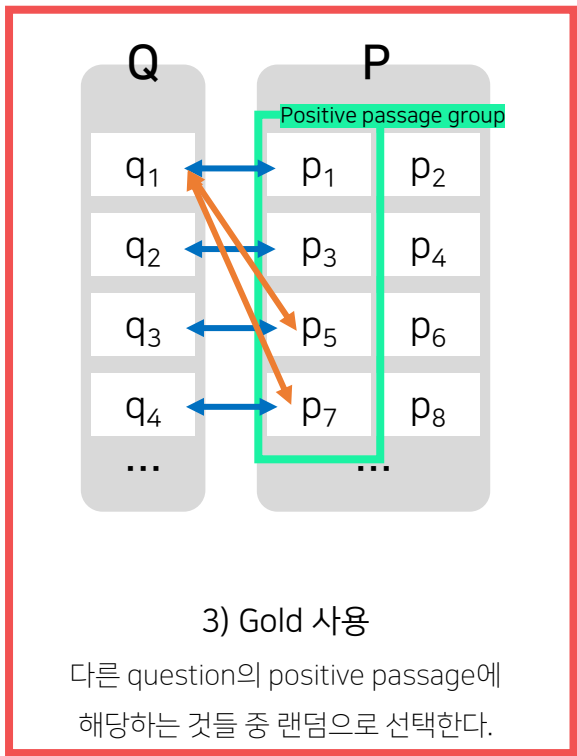
1) 랜덤 선택

Positive passage를 제외한 선택지 중
랜덤으로 선택한다.



2) BM25 기준으로 top-k 사용

정답을 포함하지 않는 passage중
BM25가 높은 것을 선택한다.



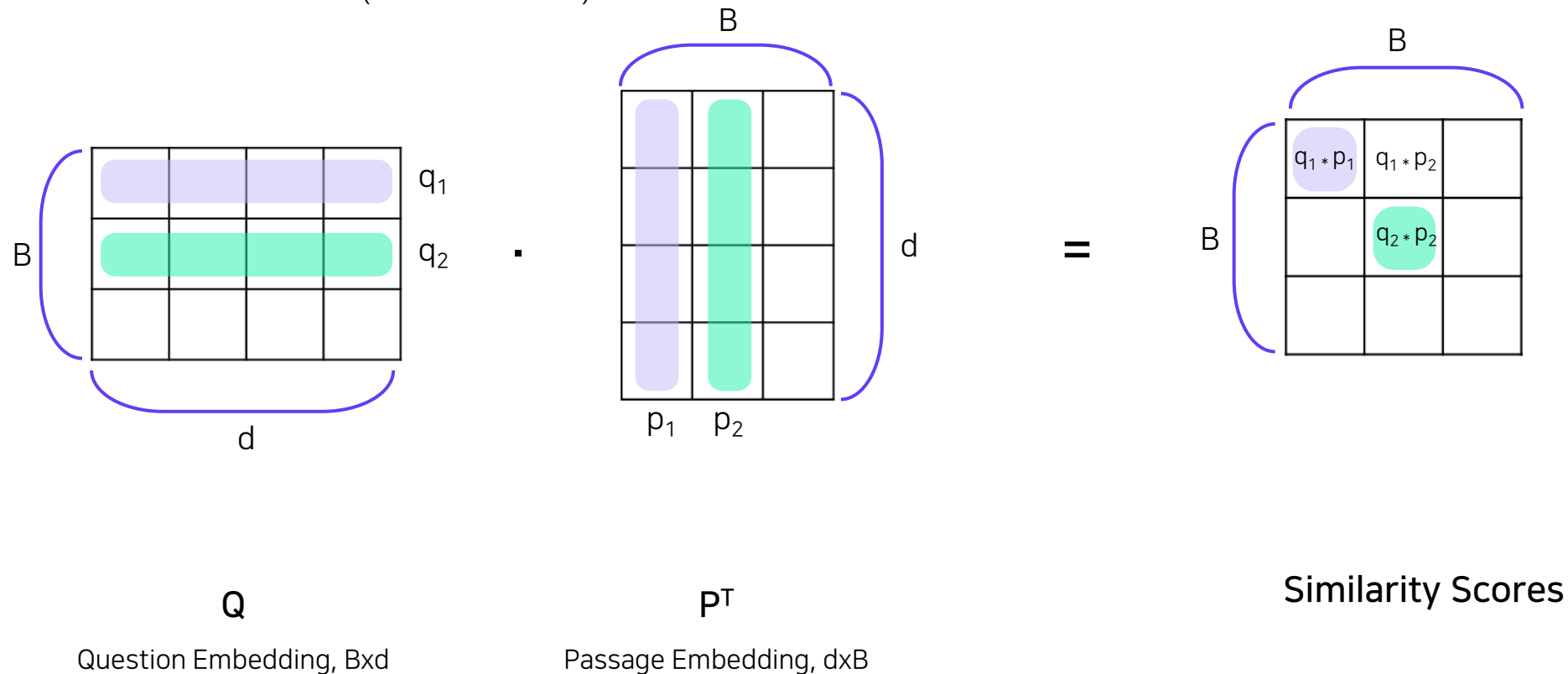
3) Gold 사용

다른 question의 positive passage에
해당하는 것들 중 랜덤으로 선택한다.

논문에서 가장 효과가 좋았던 방법: In-batch GOLD + BM25 negative sample 1개 추가

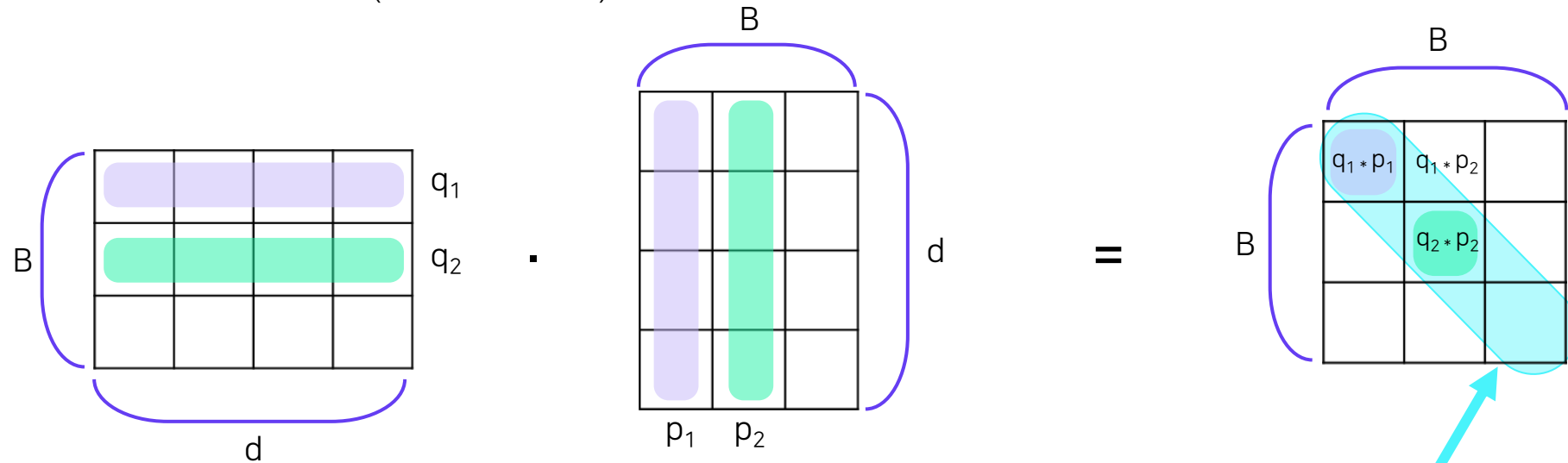
In-batch negatives

- In-batch GOLD
 - **B** : batch size, number of questions in mini-batch
 - **d** : dimension (BERT = 768)



In-batch negatives

- In-batch GOLD
 - B : batch size, number of questions in mini-batch
 - d : dimension (BERT = 768)



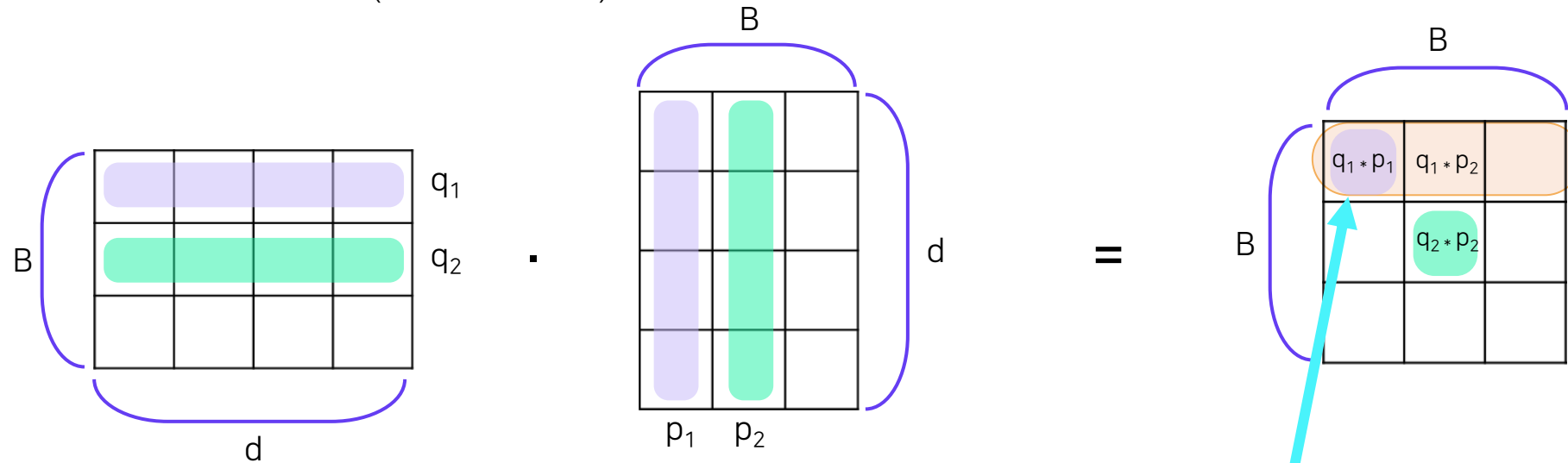
Matching되는 embedding vector ($i = j$) 의 내적: Positive examples → 최대가 되게 하는 방향으로 학습
이외: Negative examples → 최소가 되게 하는 방향으로 학습

Q
Question Embedding, $B \times d$

P
Passage Embedding, $d \times B$

In-batch negatives

- In-batch GOLD
 - B : batch size, number of questions in mini-batch
 - d : dimension (BERT = 768)



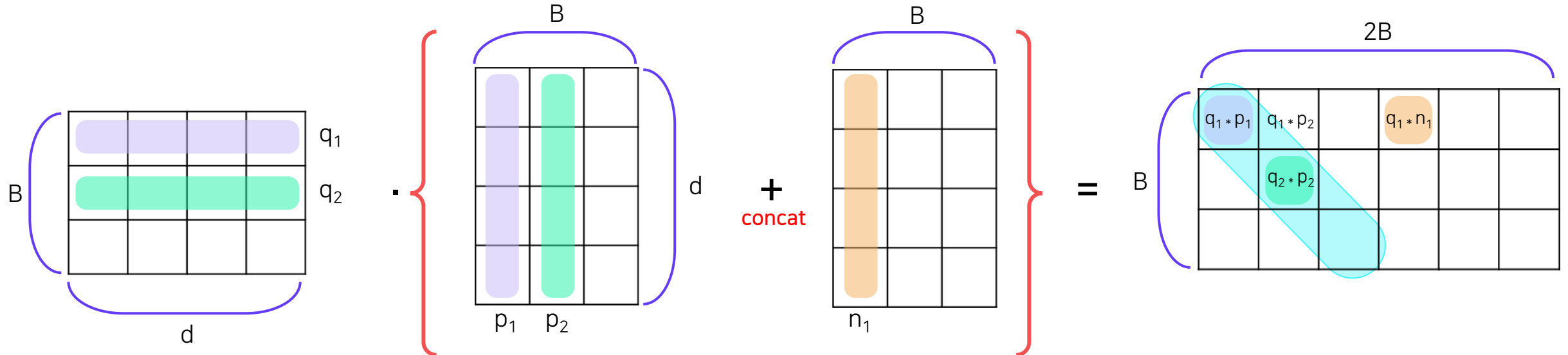
Q
Question Embedding, $B \times d$

P^T
Passage Embedding, $d \times B$

각각 Batch마다 B개의 training instances 생성
Question마다 Negative Passage B-1개

In-batch negatives

- In-batch GOLD + BM25⁽¹⁾
 - **B** : batch size, number of questions in mini-batch
 - **d** : dimension (BERT = 768)



Q

Question Embedding, Bxd

P^T

Passage Embedding, dxB

Hard negative passage

높은 BM25 점수를 가지지만 정답이 든
Passage가 아닌 경우를 negative passage로 추가
(Question마다 선택 → B개)

Similarity Scores

5. Training Dense Encoder

- $$\mathcal{D} = \{ \langle \underbrace{q_i}_{\text{Question}}, \underbrace{p_i^+}_{\text{Positive P}}, \underbrace{p_{i,1}^-, \dots, p_{i,n}^-}_{\text{Negative P}} \rangle \}_{i=1}^m$$

$$= -\log \frac{L(q_i, p_i^+, p_{i,1}^-, \dots, p_{i,n}^-) e^{\text{sim}(q_i, p_i^+)}}{e^{\text{sim}(q_i, p_i^+)} + \sum_{j=1}^n e^{\text{sim}(q_i, p_{i,j}^-)}}$$

32

Objective function

5. Training Dense Encoder

- Positive passage에 대한 Negative Log Likelihood (NLL) loss 사용

$$\mathcal{D} = \{ \langle \underbrace{q_i}_{\text{Question}}, \underbrace{p_i^+}_{\text{Positive P}}, \underbrace{p_{i,1}^-, \dots, p_{i,n}^-}_{\text{Negative P}} \rangle \}_{i=1}^m$$

m개의 passage에 대한 training data

$$L(q_i, p_i^+, p_{i,1}^-, \dots, p_{i,n}^-) = -\log \frac{e^{\text{sim}(q_i, p_i^+)}}{e^{\text{sim}(q_i, p_i^+)} + \sum_{j=1}^n e^{\text{sim}(q_i, p_{i,j}^-)}}$$

Positive passage의 score를 확률화 하기 위하여

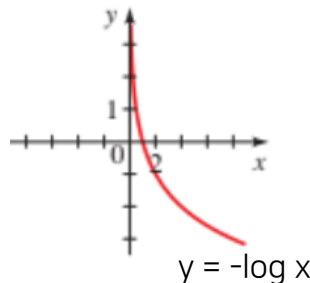
1) Positive passage와 question과의 similarity score과 negative sample의 점수를 가져와서

2) Softmax를 해서

$$\frac{e^{x_i}}{\sum_{j=1}^J e^{x_j}} \quad \text{for } i = 1, \dots, J$$

이 확률값을

3) NLL loss에 적용하여 학습



- 정답 여부 뿐 아니라 확률까지 고려한다.
- 확률 값을 음의 log함수에 넣어서 반환
- 잘못 예측할수록(실제 정답에 대한 확률이 낮아질수록) loss값이 커진다.

- **DPR Model used in experiments**
 - In-batch negative setting + 1개의 BM25 negative passage(per question)
 - Batch size: 128
 - Epoch
 - Large dataset(Natural Question, TriviaQA, SQuAD): 40
 - Small dataset(Curated TREC, WebQuestions): 100
 - Learning rate: 10^{-5}
 - Adam
 - linear scheduling with warm-up
 - Dropout rate 0.1

- Training data
 - **Single:** 1가지 Data로 훈련, 그 Data로 평가
 - ex) SQuAD dataset으로 훈련, SQuAD dataset으로 평가
 - **Multi:** SQuAD 제외한 Data(NQ, TriviaQA, TREC, WQ)로 학습, 단일 Data 평가
 - ex) 4가지 dataset으로 훈련, SQuAD dataset을 평가
 - 특정 data가 아니라 모든 데이터에 공통적으로 잘 작동하는 retriever를 만들자는 목표
- 성능 평가
 - DPR
 - BM25
 - BM25+DPR
 - BM25, DPR 각각 top-2000을 rerank
 - $\text{BM25}(q,p) + \lambda \cdot \text{sim}(q,p), \quad \lambda = 1.1$

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.

- 1) SQuAD를 제외하고 DPR이 BM25보다 성능이 좋다
- 2) K가 작을수록 DPR-BM25간의 성능 GAP이 많이 난다.
- 3) Multi로 training 시킨 것에서는 dataset의 크기가 가장 작은 Curated TREC의 성능이 가장 좋다.
Natural Question, WebQuestions는 살짝 상승, TrivaQA는 살짝 하락 (vs single DPR)
- 4) DPR+BM25의 경우, 일부의 경우에만 성능이 더 좋았다.

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

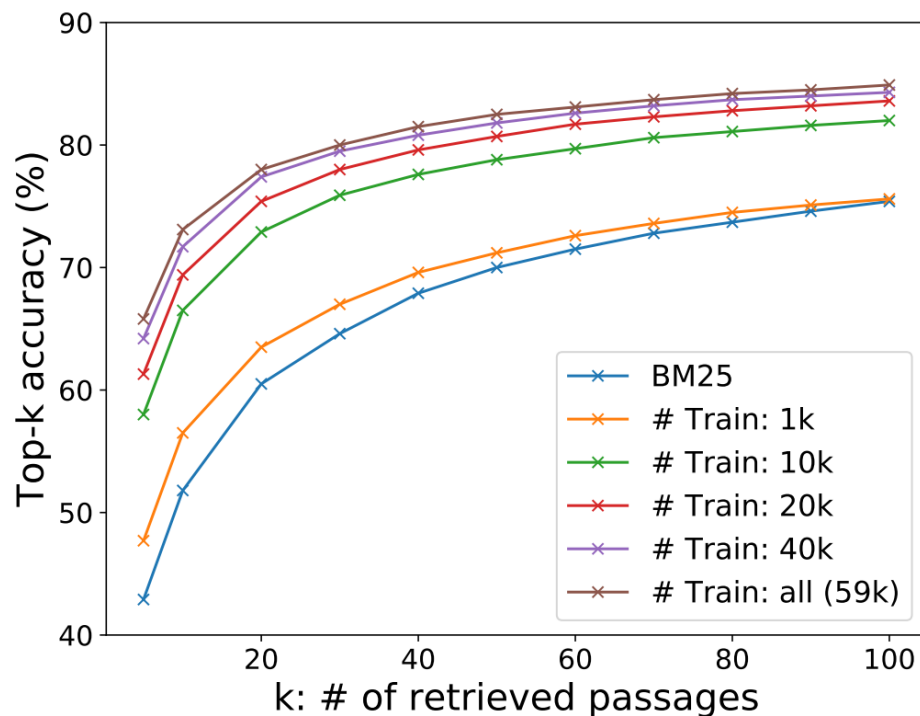
Table 2: Top-20 & Top-100 retrieval accuracy on test sets, measured as the percentage of top 20/100 retrieved passages that contain the answer. *Single* and *Multi* denote that our Dense Passage Retriever (DPR) was trained using individual or combined training datasets (all the datasets excluding SQuAD).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

- SQuAD의 경우 BM25 성능 > DPR 성능

- 1) 본문을 본 다음 질문을 작성함 → passage와 question간의 어휘 중복이 발생하여 BM25의 성능이 더 좋았을 것이다.
- 2) Wiki data로만 구성됨 → 학습 data의 분포 자체가 이미 bias 되어있음

Ablation Study on Model Training

- Sample efficiency



- 1) 1000개로만 학습시킨 것이 BM25의 성능을 능가
→ 적은 QA 쌍으로도 높은 Quality의 Dense Retriever를 학습시킬 수 있다.
- 2) Data를 더 많이 학습시킬 수록 성능은 계속 올라감

- NQ로 학습시킨 결과임
- Top-k retrieval accuracy: retrieve 된 passage 중에 답을 포함하는 passage의 비율

Ablation Study on Model Training

5. Training Dense Encoder

- In-batch negative training

- 2) In-batch 사용했을 때가 더 성능 좋다.
- 3) Batch size 따라서 성능이 지속적으로 향상된다 (=negative sample의 개수, #N)

Type	#N	IB	Top-5	Top-20	Top-100
Random	7	✗	47.0	64.3	77.8
BM25	7	✗	50.0	63.3	74.8
Gold	7	✗	42.6	63.1	78.3
Gold	7	✓	51.1	69.1	80.8
Gold	31	✓	52.1	70.8	82.1
Gold	127	✓	55.8	73.0	83.1
G.+BM25 ⁽¹⁾	31+32	✓	65.0	77.3	84.4
G.+BM25 ⁽²⁾	31+64	✓	64.5	76.4	84.0
G.+BM25 ⁽¹⁾	127+128	✓	65.8	78.0	84.9

1) $k \geq 20$ 일때는 3개의 차이가 크지 않다.

4) Hard negative passage를 1개 추가했을 때 성능이 가장 좋았음

Table 3: Comparison of different training schemes, measured as top- k retrieval accuracy on **Natural Questions** (development set). #N: number of negative examples, IB: in-batch training. G.+BM25⁽¹⁾ and G.+BM25⁽²⁾ denote in-batch training with 1 or 2 additional BM25 negatives, which serve as negative passages for all questions in the batch.

- 기타 실험 (In Appendix, 성능 개선 안 된 것들)

- 1) Impact of gold passages

- NQ Data로 실험 (Question, Answer, Passage 있음)
 - 일부 Data에는 Passage가 없고 Question, Answer만 있기 때문에
gold(positive) passage가 있는 것 vs passage를 만들어 줘야하는 Data의 성능을 비교하기 위함
 - QA를 Query로 묶고 답변이 포함된 passage중에서 BM25를 해서 top k 뽑음
 - Top-1, 5, 20, 100 각각 1점 하락

- 2) Similarity, loss

- Dot product > L2 >> cosine
 - NLL > triplet

<참고>

질문-답변만 있는 데이터셋 (CuratedTREC, WebQuestions, WikiMovies)에서 MRC 학습 데이터 만들기. Supporting document가 필요함

1. 위키피디아에서 Retriever를 이용해 관련성 높은 문서를 검색 - QA를 Query로 묶고 답변이 포함된 passage중에서 BM25를 해서 top k 뽑음
2. 너무 짧거나 긴 문서, 질문의 고유명사를 포함하지 않는 등 부적합한 문서 제거
3. answer가 exact match로 들어있지 않은 문서 제거
4. 남은 문서 중에 질문과 (사용 단어 기준) 연관성이 가장 높은 단락을 supporting evidence로 사용함

- 기타 실험 (In Appendix, 성능 개선 안 된 것들)

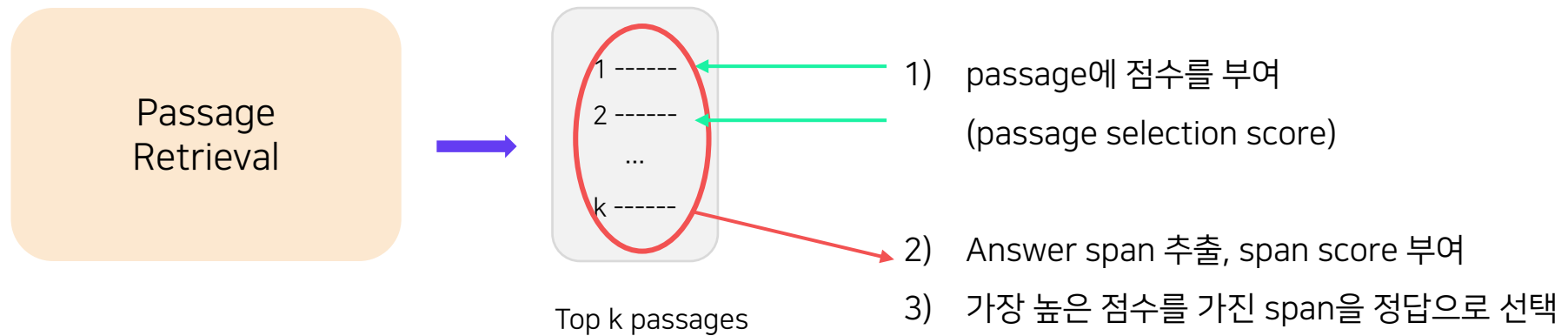
3) Cross-dataset generalization

- train- test를 다른 dataset으로 함 (train: NQ data, test: WQ, TREC dataset (large → small))
- 각각의 data로 학습시킨 모델보다 3~5점 하락
- 그럼에도 불구하고, BM25보다 성능 좋음

4) Qualitative Analysis

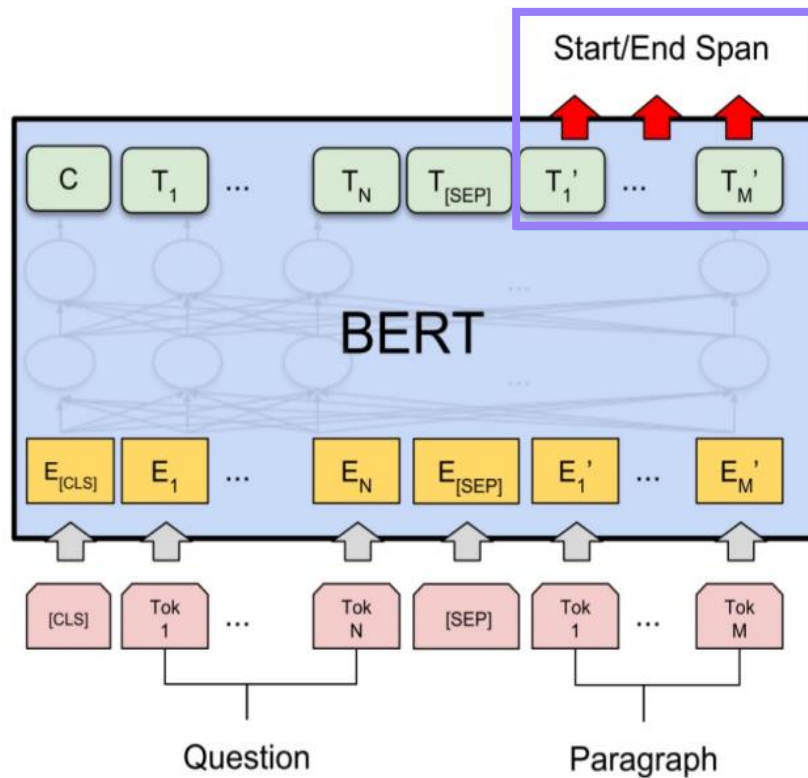
- 일반적으로 DPR성능 > BM25성능이지만, 두 방법으로 검색한 구절은 다름
 - DPR: 어휘 변형이나 의미론적 관계를 잘 찾는다. 두드러진 구(특정 단어)를 찾는 데는 약하다.
 - BM25: Term-matching에 강하다. 특정 구가 중요한 경우 사용하면 좋음

6. End-to-end QA System



- Single passage training
- Positive passage 1개, negative m-1개. 논문에서 m=24

- Reader



Output

- 각 token이 답의 시작 token일 확률
- 각 token이 답의 끝 token일 확률

Fine-tuning

- 실제 답의 start/end위치와 cross-entropy loss

Select answer

- $P_{\text{start}} \times P_{\text{end}}$ 큰 것 선택

Training	Model	NQ	TriviaQA	WQ	TREC	SQuAD
Single	BM25+BERT (Lee et al., 2019)	26.5	47.1	17.7	21.3	33.2
Single	ORQA (Lee et al., 2019)	33.3	45.0	36.4	30.1	20.2
Single	HardEM (Min et al., 2019a)	28.1	50.9	-	-	-
Single	GraphRetriever (Min et al., 2019b)	34.5	56.0	36.4	-	-
Single	PathRetriever (Asai et al., 2020)	32.6	-	-	-	56.5
Single	REALM _{Wiki} (Guu et al., 2020)	39.2	-	40.2	46.8	-
Single	REALM _{News} (Guu et al., 2020)	40.4	-	40.7	42.9	-
Single	BM25	32.6	52.4	29.9	24.9	38.1
	DPR	41.5	56.8	34.6	25.9	29.8
	BM25+DPR	39.0	57.0	35.2	28.0	36.7
Multi	DPR	41.5	56.8	42.4	49.4	24.1
	BM25+DPR	38.8	57.9	41.1	50.6	35.8

Table 4: End-to-end QA (Exact Match) Accuracy. The first block of results are copied from their cited papers. REALM_{Wiki} and REALM_{News} are the same model but pretrained on Wikipedia and CC-News, respectively. *Single* and *Multi* denote that our Dense Passage Retriever (DPR) is trained using individual or combined training datasets (all except SQuAD). For WQ and TREC in the *Multi* setting, we fine-tune the reader trained on NQ.

- 1) SQuAD를 제외하고 retriever 성능이 DPR > BM25
- 2) 큰 Data(NQ, TrivaQA)의 경우 end-to-end 성능이 multi와 single에 큰 차이가 없다.
작은 Data(WQ, TREC)의 경우에는 DPR의 multi data 학습이 성능이 더 좋다.
- 3) 추가 훈련과 복잡한 end-to-end 훈련을 하는 ORQA, REALM보다 성능이 더 좋다.
Target training set이 작은 경우(WQ, TREC) Multi Training의 성능이 더 좋다.

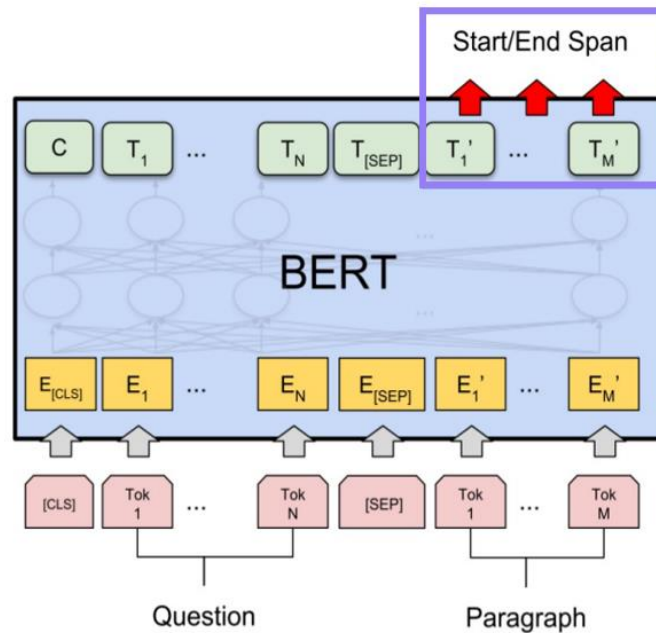
End to end 학습을 하지 않은 DPR의 Reader도
Retriever가 반환한 질 좋은 문서를 받는다면 ODQA의 성능을 높일 수 있다.

QnA

1. $P_{\text{start}} \times P_{\text{end}}$ 를 하는 이유?

p. 44

- Reader



Output

- 각 token이 답의 시작 token일 확률
- 각 token이 답의 끝 token일 확률

Fine-tuning

- 실제 답의 start/end위치와 cross-entropy loss

Select answer

- $P_{\text{start}} \times P_{\text{end}}$ 큰 것 선택

1. $P_{start} \times P_{end}$ 를 하는 이유?

- 어제 모든 토큰에 대하여 start, end 토큰일 확률을 구한다고 한다고 했는데, $k=10$ 일경우 start 큰순으로 10개, end 큰순으로 10개를 뽑습니다.

Start			End		
token_num	P_start		token_num	P_end	
7	0.97		8	0.92	
3	0.95		12	0.85	
15	0.88		40	0.81	
23	0.78		37	0.79	
18	0.72		21	0.76	
6	0.71		31	0.6	
1	0.7		15	0.57	
30	0.52		11	0.51	
26	0.5		27	0.47	
11	0.48		22	0.45	

1. $P_{start} \times P_{end}$ 를 하는 이유?

- 그러면 (start, end)의 순서쌍이 $10 \times 10 \rightarrow 100$ 개가 나오게 됩니다. 이때 start, end가 불가능할 조합일 경우를 제외합니다.

Start	End
token_num	token_num
16	8
16	12
16	40
16	37
16	21
16	31
16	15
16	11
16	27
16	22
3	8
3	12
3	40
3	37
3	21
3	31
3	15
3	11
3	27
3	22
15	8
15	12
15	40

...

- end position > start position인 경우
- question쪽에 정답이 나온 경우
- max_answer_length보다 길이가 긴 경우 등...

이런 경우, 시작 토큰 위치가
끝나는 토큰 위치보다 뒤에 있으므로
불가능한 경우임

1. $P_{start} \times P_{end}$ 를 하는 이유?

- 이때 start, end쌍을 재정렬하는 과정에서 곱하기를 사용합니다!

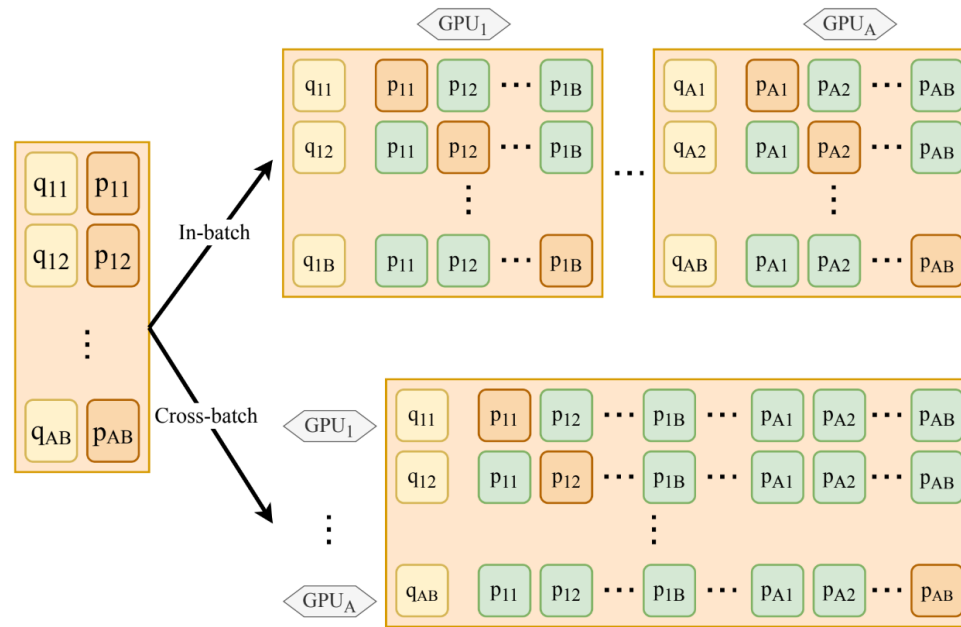
Start	End	
token_num	token_num	
16	8	
16	12	
16	40	0.7857
16	37	0.7663
16	21	0.7372
16	31	0.582
16	15	
16	11	
16	27	0.4559
16	22	0.4365
3	8	0.874
3	12	0.8075
3	40	0.7695
3	37	0.7505
3	21	0.722
3	31	0.57
3	15	0.5415

...

'가능한' 쌍에 대한 정렬 기준으로 사용

요약: start와 end를 묶어서 표현할 지표가 필요하기 때문

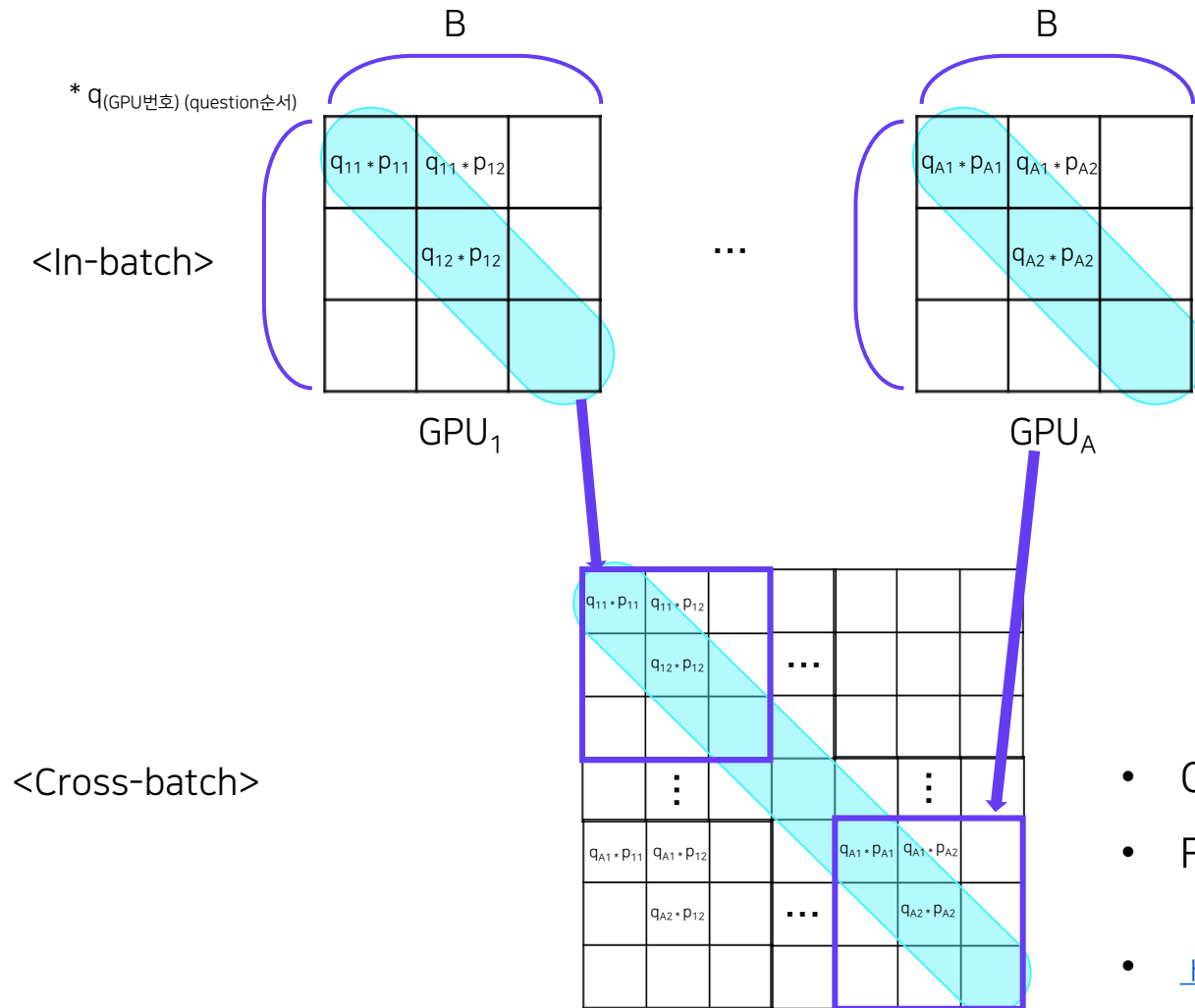
2. cross-batch negative sampling



- RocketQA (<https://arxiv.org/pdf/2010.08191.pdf>)에서 제시한 방법
- 더 많은 negative sample을 이용하여 학습시키기 위함

2. cross-batch negative sampling

QnA



- Question마다 $A \times B - 1$ 개의 negative passage를 보게 됨
- FleetX이라는 분산 훈련 엔진에서 구현됨
 - 모든 GPU를 통틀어 연산이 가능
- <https://github.com/PaddlePaddle/Research/tree/master/NLP/NAACL2021-RocketQA>
- <https://github.com/PaddlePaddle/FleetX>

Reference

- [https://ko.wikipedia.org/wiki](https://ko.wikipedia.org/wiki/acl2020-openqa-tutorial)
- [acl2020-openqa-tutorial](#)