CSCE 670 - Information Storage and Retrieval

Lecture 18: BERT-Based Ranking, ColBERT

Yu Zhang

yuzhang@tamu.edu

October 28, 2025

Course Website: https://yuzhang-teaching.github.io/CSCE670-F25.html

Adapted from the slides by Prof. James Caverlee

Unfinished Part from Last Lecture

Recap: Transformer as a Black Box

- Transformer is a neural network
- It has two types of architecture: encoder and decoder
- BERT uses the Transformer encoder architecture

- Input to a Transformer encoder can be a piece of text:
  - A sequence of words $w_1, w_2, \dots, w_L$
  - Represented by their corresponding embeddings $\boldsymbol{e}_{w_1}, \boldsymbol{e}_{w_2}, \dots, \boldsymbol{e}_{w_L}$

- Then, the output is a sequence of contextualized word vectors $\boldsymbol{h}_{w_1}, \boldsymbol{h}_{w_2}, \dots, \boldsymbol{h}_{w_L}$
  - The output vector $\boldsymbol{h}_{w_i}$ captures the meaning of w_i by considering the entire input sequence as w_i's context
  - $\boldsymbol{h}_{w_i} = \text{Transformer}(\boldsymbol{e}_{w_i} | \boldsymbol{e}_{w_1}, \boldsymbol{e}_{w_2}, \dots, \boldsymbol{e}_{w_L})$

BERT [Devlin et al., NAACL 2019]

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

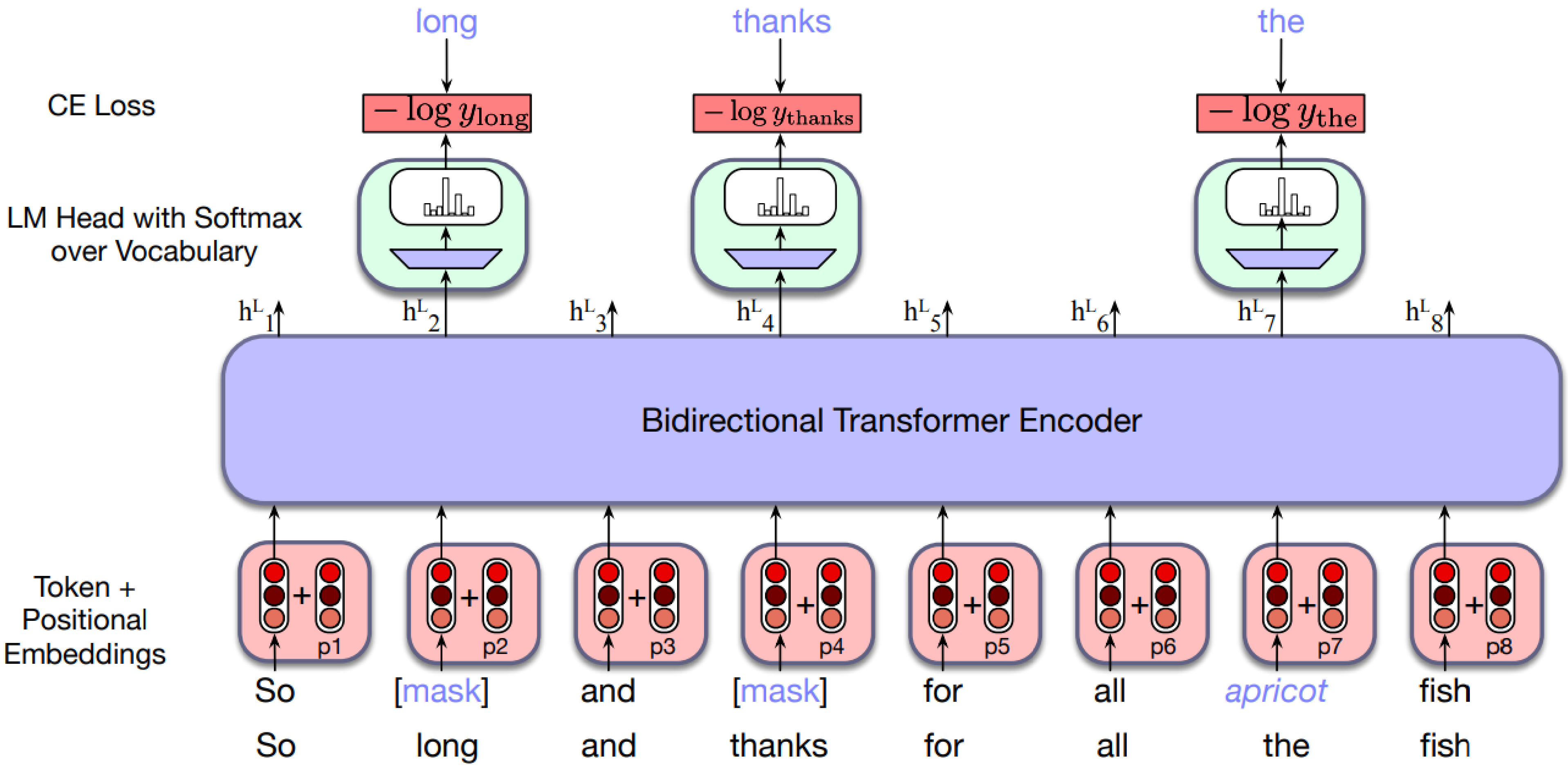
Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both

There are two existin
ing pre-trained language r
stream tasks: *feature-base*
feature-based approach,
et al., 2018a), uses task-sp
include the pre-trained r
tional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal

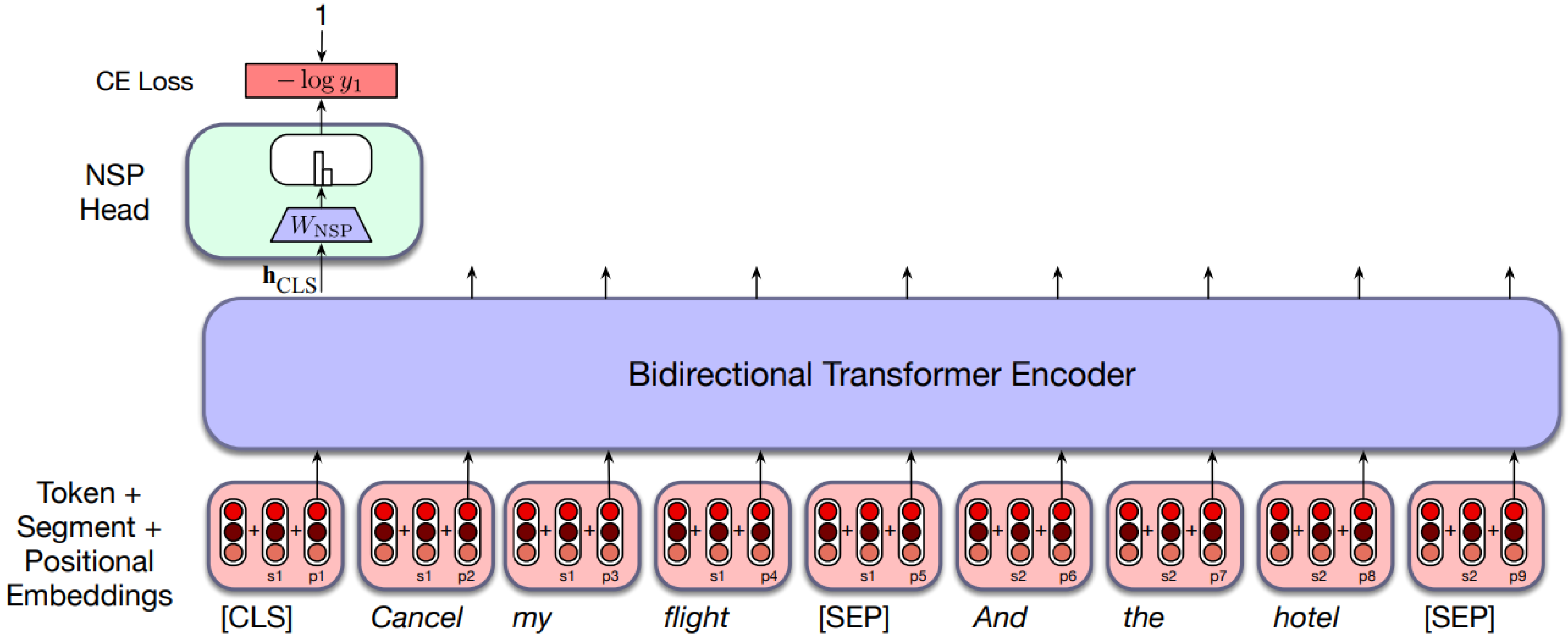
BERT Pre-training

- Task 1 – Masked Language Modeling (MLM): With 15% words randomly masked, the model learns bidirectional contextual information to predict the masked words.



BERT Pre-training

- Task 2 – Next Sentence Prediction (NSP): The model is presented with pairs of sentences. It is trained to predict whether each pair consists of an actual pair of adjacent sentences from the training corpus or a pair of unrelated sentence.



Source: https://web.stanford.edu/~jurafsky/slp3/10.pdf

Immediate Impact of BERT

- In 2018, BERT came out and largely outperformed most previous methods on common NLP tasks (e.g., sentiment classification, natural language inference, question answering).
- BERT got the best paper award at the NAACL 2019 conference.

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\mathrm{BERT}_{\mathrm{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\mathrm{BERT}_{\mathrm{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

- “Open AI GPT”: GPT-2
- “BERT-Base”: 12 Transformer encoder layers; ~110M parameters
- “BERT-Large”: 24 Transformer encoder layers; ~340M parameters

Improving BERT: RoBERTa [Liu et al., arXiv 2019]

- Next Sentence Prediction (NSP) is not helpful!
- Only use Masked Language Modeling (MLM)

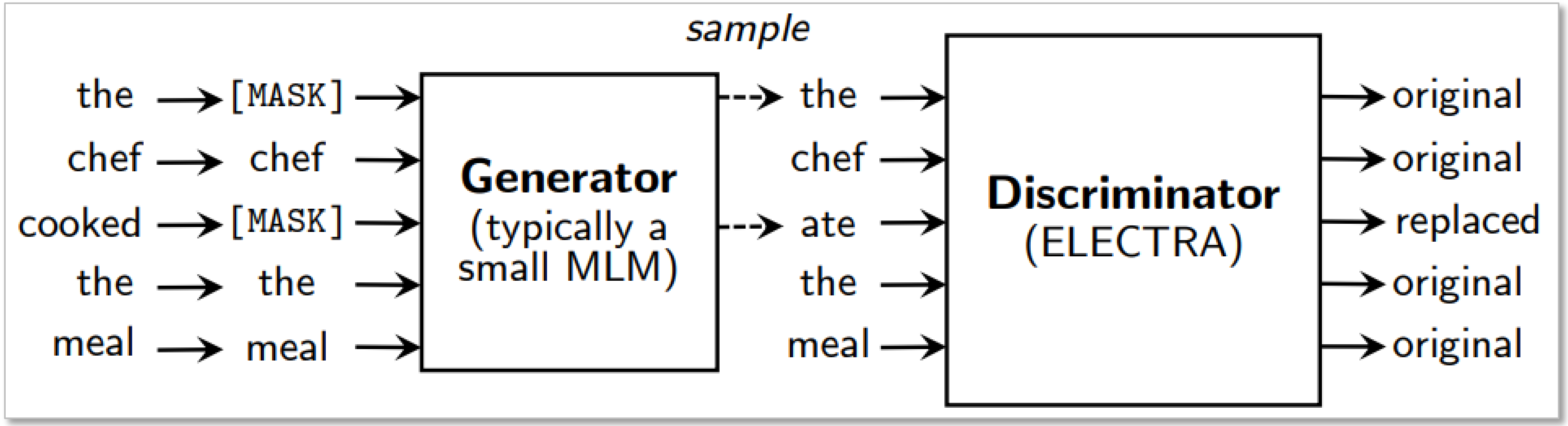
Model	**SQuAD 1.1/2.0**	**MNLI-m**	**SST-2**	**RACE**
Our reimplementation (with NSP loss):				
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0
Our reimplementation (without NSP loss):				
FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8
DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6

- Pretrain on longer sequences
- Pretrain the model for longer, with bigger batches
- Pretrain over more data
- Dynamically change the masking patterns applied to the training data in each epoch

Model	**data**	**bsz**	**steps**	**SQuAD** (v1.1/2.0)	**MNLI-m**	**SST-2**
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**

Improving BERT: ELECTRA [Clark et al., ICLR 2020]

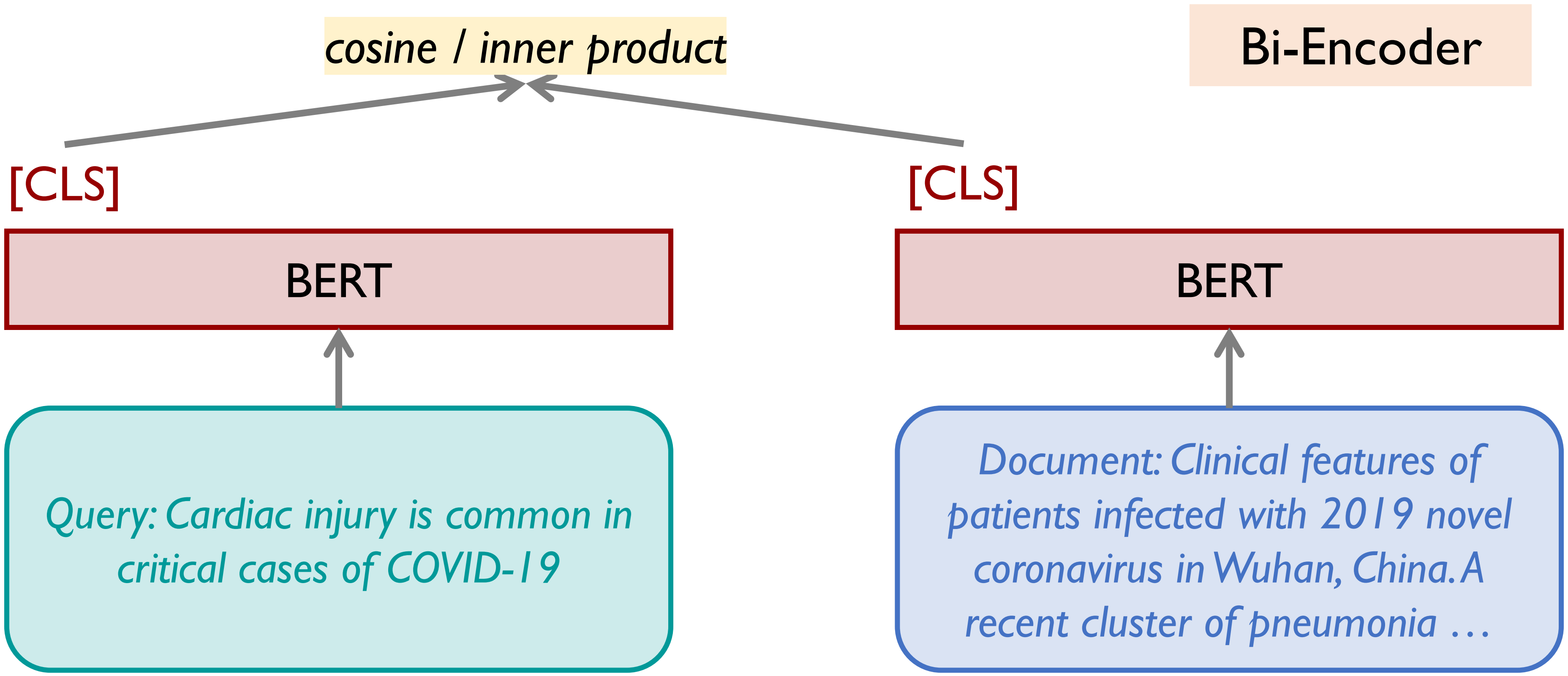
- Use a small MLM as an auxiliary generator (discarded after pretraining)
- Pretrain the main model as a discriminator
- The small auxiliary MLM and the main discriminator are jointly trained.
- The main model's pretraining task becomes more and more challenging in pretraining.



Content of This Lecture

How to use BERT for retrieval? – Solution 1

- Encode query and document separately
- The output vector of the [CLS] token serves as query / document embedding



Python Implementation to Encode a Query / Document

python

Copy code

```python
from transformers import BertTokenizer, BertModel
import torch

# Load pre-trained BERT-base model and tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')

# Input text
text = "Cardiac injury is common in critical cases of COVID-19"

# Tokenize and encode input
inputs = tokenizer(text, return_tensors='pt', truncation=True, padding=True)

# Forward pass (no gradient needed)
with torch.no_grad():
    outputs = model(**inputs)

# Extract [CLS] token embedding
cls_embedding = outputs.last_hidden_state[:, 0, :]  # shape: [1, 768]
```

Fine-Tune a Bi-Encoder Ranking Model

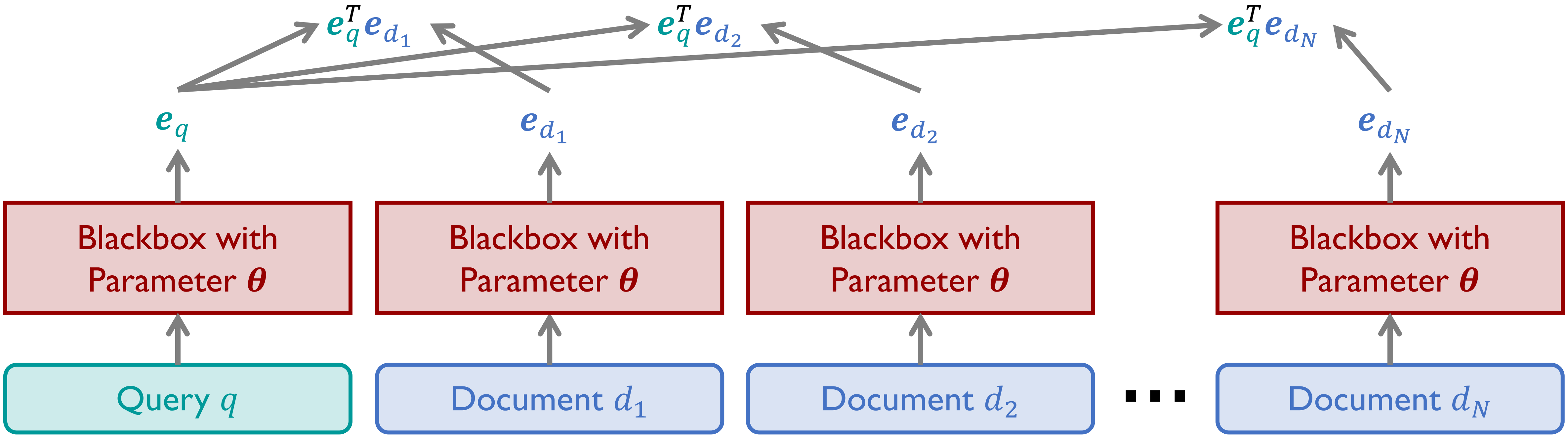
- Fine-tuning: continue to train a pre-trained model using supervised learning on labeled data for a specific downstream task

- What if I want to train a Bi-Encoder using learning to rank?
  - Parameters: All parameters in the query Transformer encoder and the document Transformer encoder
    - At the beginning of training, both encoders are initialized with BERT
    - In many Bi-Encoder ranking models, the query encoder and the document encoder share all parameters
  - Learning Objective?
    - Recall Dense Passage Retrieval!

Dense Passage Retrieval [Karpukhin et al., EMNLP 2020]

- Learning objective: Given a query q and N documents $(d_1, d_2, \ldots, d_N)$, the ground truth tells us that d_1 is the most relevant to q among these documents

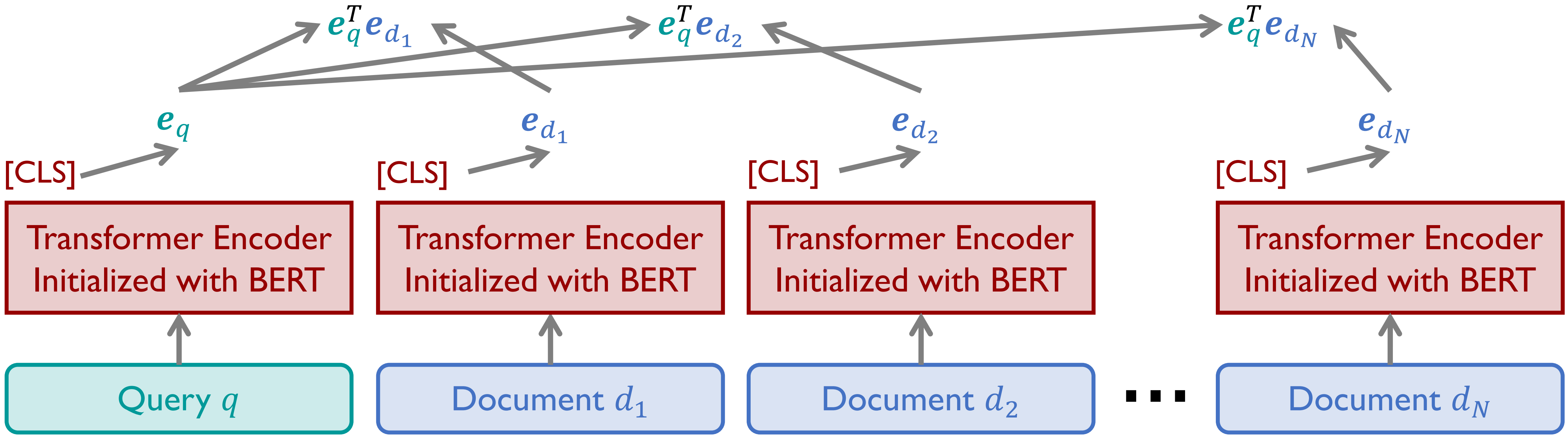
$$\frac{\exp(\boldsymbol{e}_q^T \boldsymbol{e}_{d_1})}{\sum_{i=1}^{N} \exp(\boldsymbol{e}_q^T \boldsymbol{e}_{d_i})}$$



Dense Passage Retrieval (BERT Version)

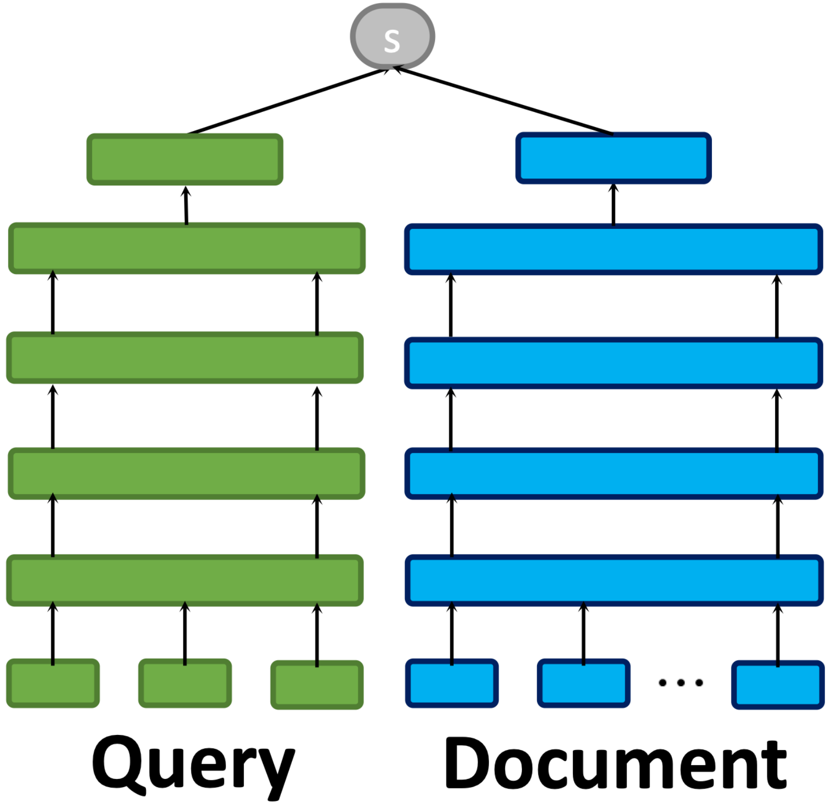
- Learning objective: Given a query q and N documents $(d_1, d_2, \ldots, d_N)$, the ground truth tells us that d_1 is the most relevant to q among these documents

$$\frac{\exp(\boldsymbol{e}_q^T \boldsymbol{e}_{d_1})}{\sum_{i=1}^{N} \exp(\boldsymbol{e}_q^T \boldsymbol{e}_{d_i})}$$



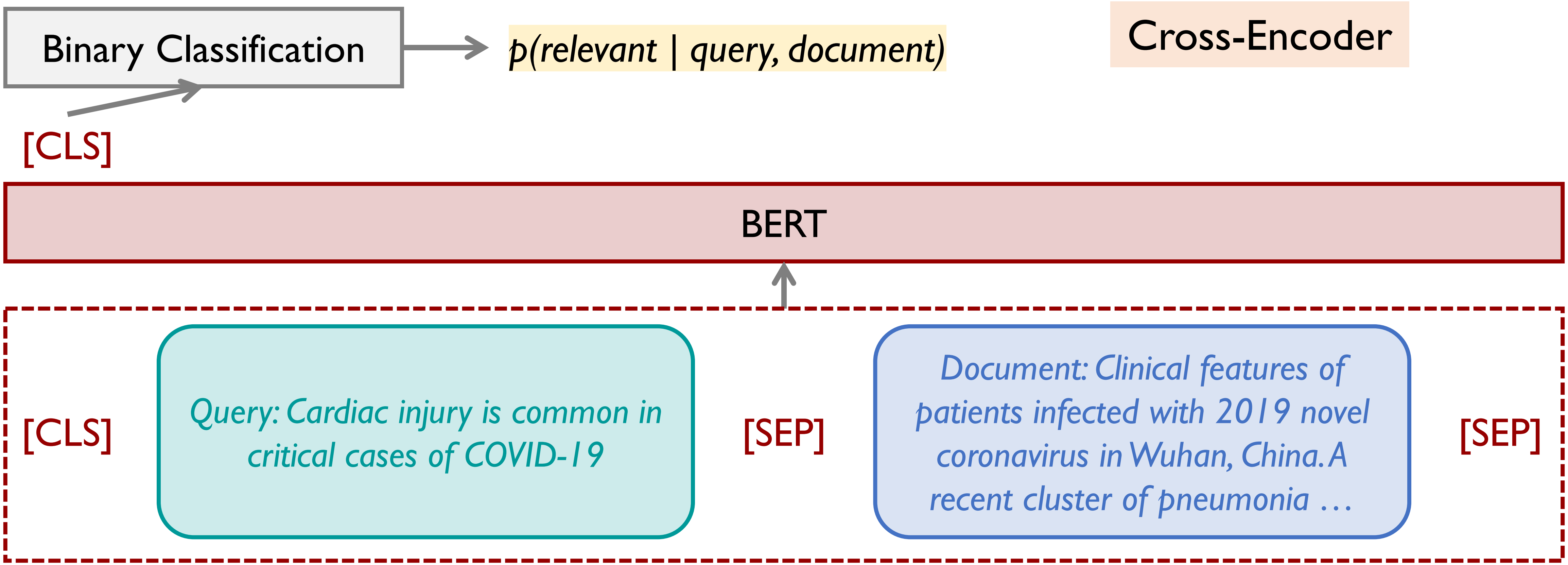
Fine-Tune a Bi-Encoder Ranking Model

- In-batch negative sampling
  - The larger the batch size, the better
- What potential issues might a Bi-Encoder have?
  - Two encoders independently encode the query and the document into vectors and calculate their similarity.
  - However, the importance of a query word may vary across different documents; the importance of a document word may also vary across different queries.



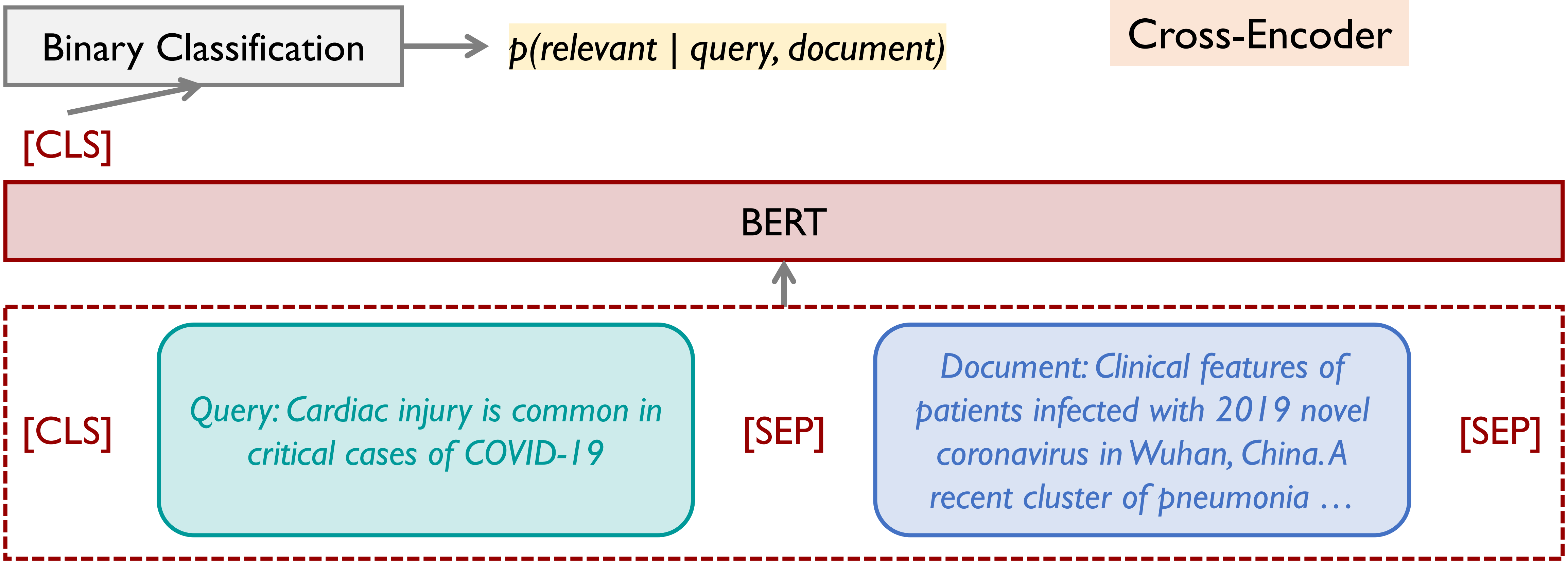
How to use BERT for retrieval? – Solution 2

- Concatenate the query and document into a single input sequence
- Get the representation of the entire sequence and perform binary classification



Fine-Tune a Cross-Encoder Ranking Model

- Parameters: All parameters in the Transformer encoder + the classification layer
- Objective: Cross-Entropy Loss, $-(y \log p + (1-y)\log(1-p))$



Cross-Encoder Ranking Model [Dai and Callan, SIGIR 2019]

Deeper Text Understanding for IR with Contextual Neural Language Modeling

Zhuyun Dai
Carnegie Mellon University
zhuyund@cs.cmu.edu

Jamie Callan
Carnegie Mellon University
callan@cs.cmu.edu

ABSTRACT

Neural networks provide new possibilities to automatically learn complex language patterns and query-document relations. Neural IR models have achieved promising results in learning query-document relevance patterns, but few explorations have been done on understanding the text content of a query or a document. This paper studies leveraging a recently-proposed contextual neural language model, BERT, to provide deeper text understanding for IR. Experimental results demonstrate that the contextual text representations from BERT are more effective than traditional word embeddings. Compared to bag-of-words retrieval models, the contextual language model can better leverage language structures, bringing large improvements on queries written in natural languages. Combining the text understanding ability with search knowledge leads

related words. But word co-occurrence is only a shallow bag-of-words understanding of the text. Recently, we have seen rapid progress in text understanding with the introduction of pre-trained neural language models such as ELMo [8] and BERT [3]. Different from traditional word embeddings, they are *contextual* – the representation of a word is a function of the entire input text, with word dependencies and sentence structures taken into consideration. The models are *pre-trained* on a large number of documents so that the contextual representations encode general language patterns. Contextual neural language models have outperformed traditional word embeddings on a variety of NLP tasks [3, 8].

The deeper text understanding of contextual neural language models brings new possibilities to IR. This paper explores leveraging BERT (Bidirectional Encoder Representations from Transform-

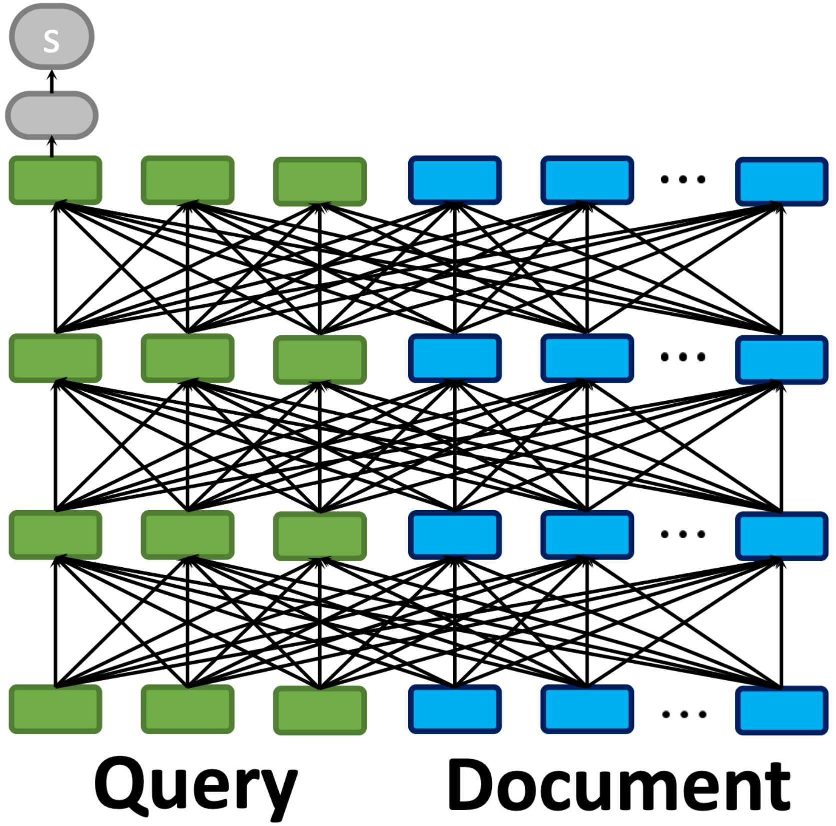
What if the document is too long?

- BERT can take at most 512 tokens
- In a Cross-Encoder architecture, the document needs to share the 512 tokens with the query
- Divide the document d into passages $p_1, p_2, \dots, p_K$
  - FirstP
    $$\text{score}(q, d) = \text{score}(q, p_1)$$
  - MaxP
    $$\text{score}(q, d) = \max_{1 \le k \le K} \text{score}(q, p_k)$$
  - SumP
    $$\text{score}(q, d) = \sum_{k=1}^{K} \text{score}(q, p_k)$$

	nDCG@20			
	Robust04		ClueWeb09-B	
Model	Title	Description	Title	Description
BOW	0.417	0.409	0.268	0.234
SDM	0.427	0.427	0.279	0.235
RankSVM	0.420	0.435	0.289	0.245
Coor-Ascent	0.427	0.441	**0.295**	0.251
DRMM	0.422	0.412	0.275	0.245
Conv-KNRM	0.416	0.406	0.270	0.242
BERT-FirstP	$0.444^{\dagger}$	$0.491^{\dagger}$	0.286	$\mathbf{0.272}^{\dagger}$
BERT-MaxP	$\mathbf{0.469}^{\dagger}$	$\mathbf{0.529}^{\dagger}$	0.293	$0.262^{\dagger}$
BERT-SumP	$0.467^{\dagger}$	$0.524^{\dagger}$	0.289	0.261

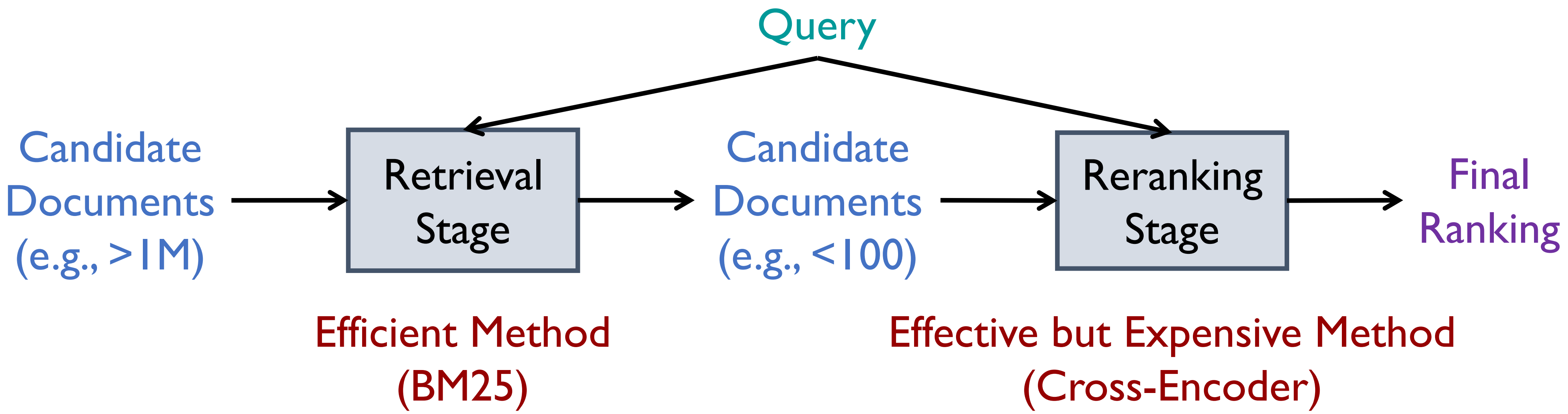
Fine-Tune a Cross-Encoder Ranking Model

- All-to-All Interaction
  - The context of each query word includes the entire query AND the entire document
  - The context of each document word includes the entire query AND the entire document
- What potential issues might a Cross-Encoder have?
  - Cross-Encoder is expensive!
  - You cannot precompute query and word representations
  - How to solve this?



Retrieval-Reranking Paradigm (BERT Version)

- We want to use effective but expensive ranking models …
- … only for a more fine-grained ranking of the most relevant documents.



Cross-Encoder Reranking Model [Nogueira and Cho, arXiv 2019]

PASSAGE RE-RANKING WITH BERT

Rodrigo Nogueira
New York University
rodrigonogueira@nyu.edu

Kyunghyun Cho
New York University
Facebook AI Research
CIFAR Azrieli Global Scholar
kyunghyun.cho@nyu.edu

ABSTRACT

Recently, neural models pretrained on a language modeling task, such as ELMo (Peters et al., 2017), OpenAI GPT (Radford et al., 2018), and BERT (Devlin et al., 2018), have achieved impressive results on various natural language processing tasks such as question-answering and natural language inference. In this paper, we describe a simple re-implementation of BERT for query-based passage re-ranking. Our system is the state of the art on the TREC-CAR dataset and the top entry in the leaderboard of the MS MARCO passage retrieval task, outperforming the previous state of the art by 27% (relative) in MRR@10. The code to reproduce our results is available at https://github.com/nyu-dl/dl4marco-bert

BERT-Based Neural Re-Ranking

Method	MS MARCO MRR@10 Dev	MS MARCO MRR@10 Eval	TREC-CAR MAP Test
BM25 (Lucene, no tuning)	16.7	16.5	12.3
BM25 (Anserini, tuned)	-	-	15.3
Co-PACRR* (MacAvaney et al., 2017)	-	-	14.8
KNRM (Xiong et al., 2017)	21.8	19.8	-
Conv-KNRM (Dai et al., 2018)	29.0	27.1	-
IRNet†	27.8	28.1	-
BERT Base	34.7	-	31.0
BERT Large	36.5	35.8	33.5

New SOTA on MS MARCO Leaderboard!

Rank	Model	Submission Date	MRR@10 On Eval
1	**BERT + Small Training** Rodrigo Nogueira and Kyunghyun Cho - New York University	January 7th, 2019	35.87
2	**IRNet (Deep CNN/IR Hybrid Network)** Dave DeBarr, Navendu Jain, Robert Sim, Justin Wang, Nirupama Chandrasekaran – Microsoft	January 2nd, 2019	28.061

MS MARCO

https://microsoft.github.io/msmarco/

MS MARCO

Home Document Ranking Passage Ranking Updates Submissions About

MS MARCO

Follow @MSMarcoAI

Starting with a paper released at NIPS 2016, MS MARCO is a collection of datasets focused on deep learning in search.

The first dataset was a question answering dataset featuring 100,000 real Bing questions and a human generated answer. Since then we released a 1,000,000 question dataset, a natural language generation dataset, a passage ranking dataset, keyphrase extraction dataset, crawling dataset, and a conversational search.

MS MARCO

- Largest public IR benchmark release in 2016
  - Used by TREC from 2019 to 2024
  - Consists of more than 500K Bing search queries
  - Sparse labels: approximately one relevance document per query!
  - Passage Ranking: 9M short passages
  - Document Ranking: 3M long documents

MS MARCO Document Ranking Leaderboard

Search:

date		description	team	paper	code	type	MRR@100 (Dev)	MRR@100 (Eval)
2022/02/08	🏆	**coCondenser(maxp) + MORES+**	Luyu Gao - Carnegie Mellon University			full ranking	0.512	0.446
2021/07/14	🏆	**UniRetriever**	Microsoft-Research-Asia and STCA-BingAdsSelection			full ranking	0.500	0.440

MRR (Mean Reciprocal Rank)

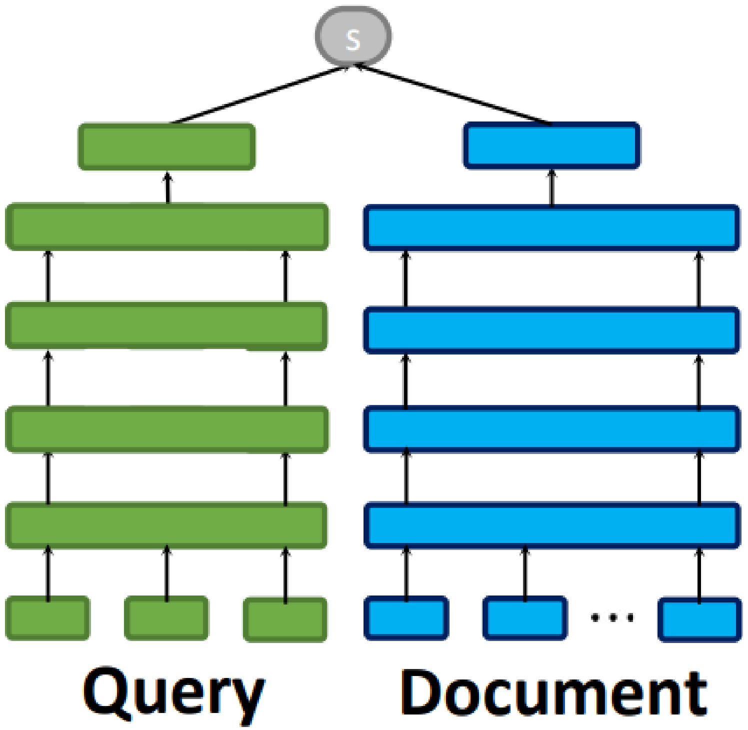
$$\mathrm{MRR} = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{\mathrm{rank}_q}$$

- Q: the set of queries
- rank_q: rank position of the first relevant result for query q

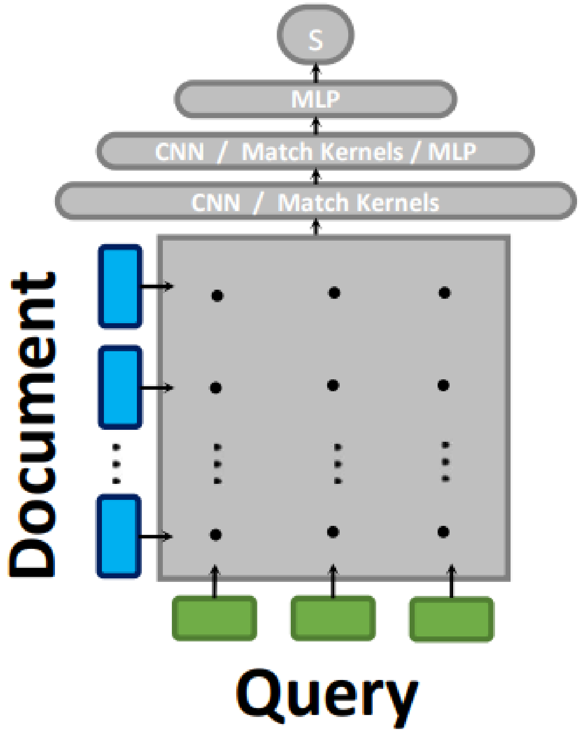
- Example
  - Suppose we have 4 candidate documents. For the following 3 queries, each query has a ranking list of the 4 documents, with the relevance ground truth as follows:
  - Query q_1: [0, 0, 0, 1]
  - Query q_2: [1, 1, 0, 1]
  - Query q_3: [0, 1, 1, 1]
  - $\mathrm{MRR} = \frac{1}{3} \times \left(\frac{1}{4} + \frac{1}{1} + \frac{1}{2}\right) = \frac{7}{12} \approx 0.583$

Questions?

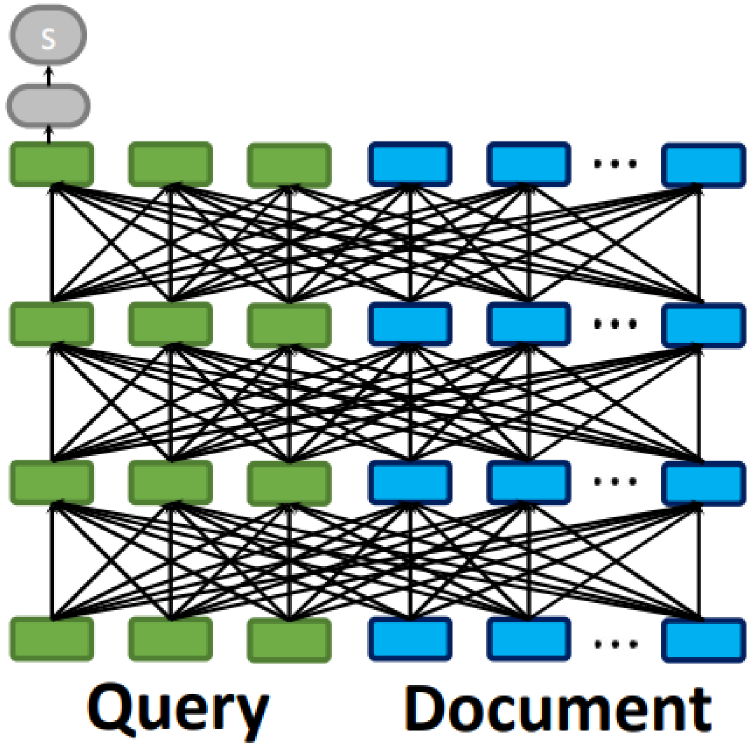
Previously Introduced Neural Ranking Paradigms



(a) Representation-based Similarity



(b) Query-Document Interaction



(c) All-to-all Interaction

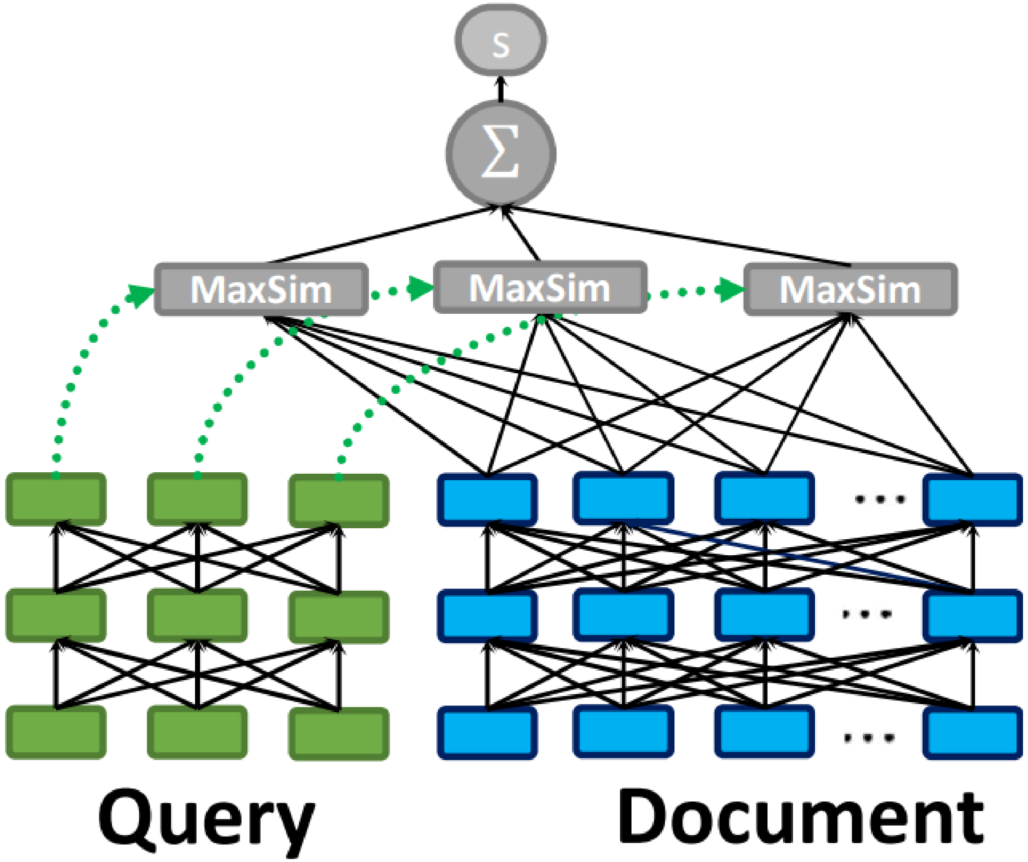
- DESM, DPR, BERT Bi-Encoder
- Efficient
- Independent Query/Doc encoding

- Duet, Conv-KNRM
- Effective but expensive
- context(Query) = Doc

- BERT Cross-Encoder
- Effective but expensive
- context(Query) = Query & Doc

Late Interaction

- Can we keep precomputation and still have fine-grained query-document interactions?
- Desired Properties:
  - Independent encoding
  - Fine-grained representations
  - End-to-end retrieval (the retrieval-reranking paradigm can be used, but it is not mandatory)



(d) Late Interaction

ColBERT [Khattab and Zaharia, SIGIR 2020]

ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT

Omar Khattab
Stanford University
okhattab@stanford.edu

Matei Zaharia
Stanford University
matei@cs.stanford.edu

ABSTRACT

Recent progress in Natural Language Understanding (NLU) is driving fast-paced advances in Information Retrieval (IR), largely owed to fine-tuning deep language models (LMs) for document ranking. While remarkably effective, the ranking models based on these LMs increase computational cost by orders of magnitude over prior approaches, particularly as they must feed each query–document pair through a massive neural network to compute a single relevance score. To tackle this, we present ColBERT, a novel ranking model that adapts deep LMs (in particular, BERT) for efficient retrieval. ColBERT introduces a *late interaction* architecture that independently encodes the query and the document using BERT and then employs a cheap yet powerful interaction step that models their

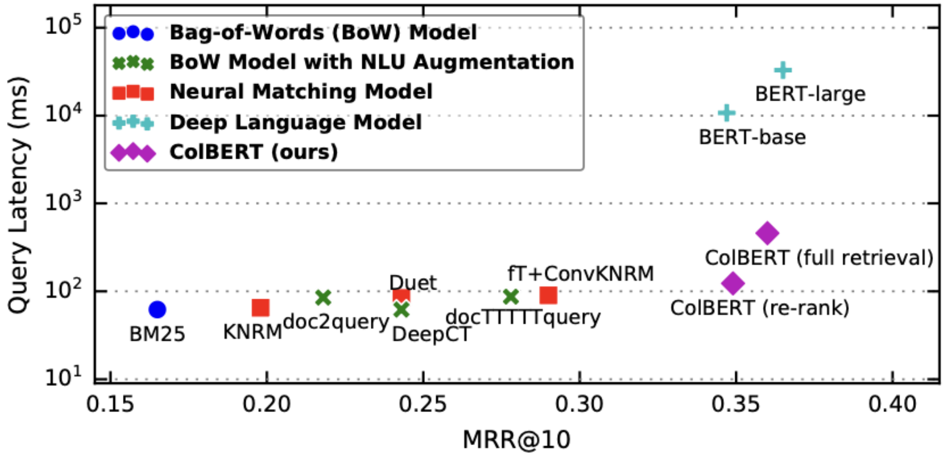


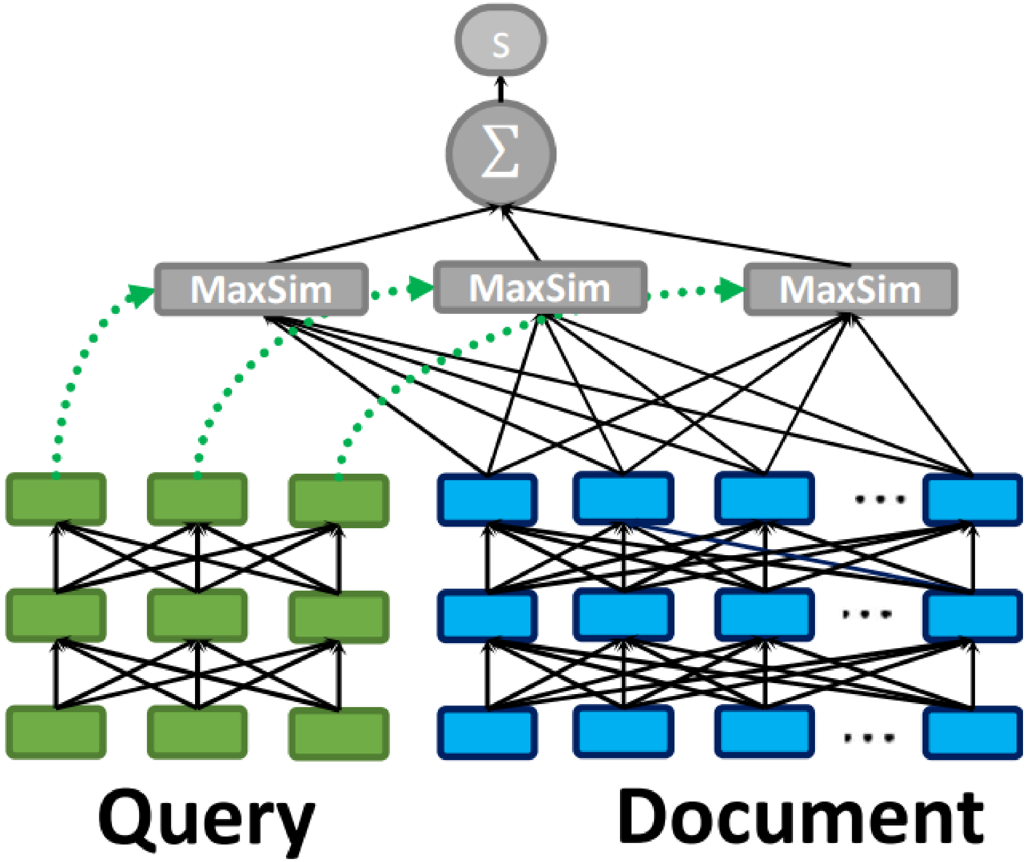
Figure 1: Effectiveness (MRR@10) versus Mean Query Latency (log-scale) for a number of representative ranking

Late Interaction

- Step 1: Encode the document into a sequence of vectors $\boldsymbol{e}_{d_1}, \boldsymbol{e}_{d_2}, \dots, \boldsymbol{e}_{d_L}$
- Step 2: Encode the query into a sequence of vectors $\boldsymbol{e}_{q_1}, \boldsymbol{e}_{q_2}, \dots, \boldsymbol{e}_{q_M}$
- Step 3:

$$\text{score}(q, d) = \sum_{m=1}^{M} \max_{1 \le l \le L} \boldsymbol{e}_{q_m}^{T} \boldsymbol{e}_{d_l}$$

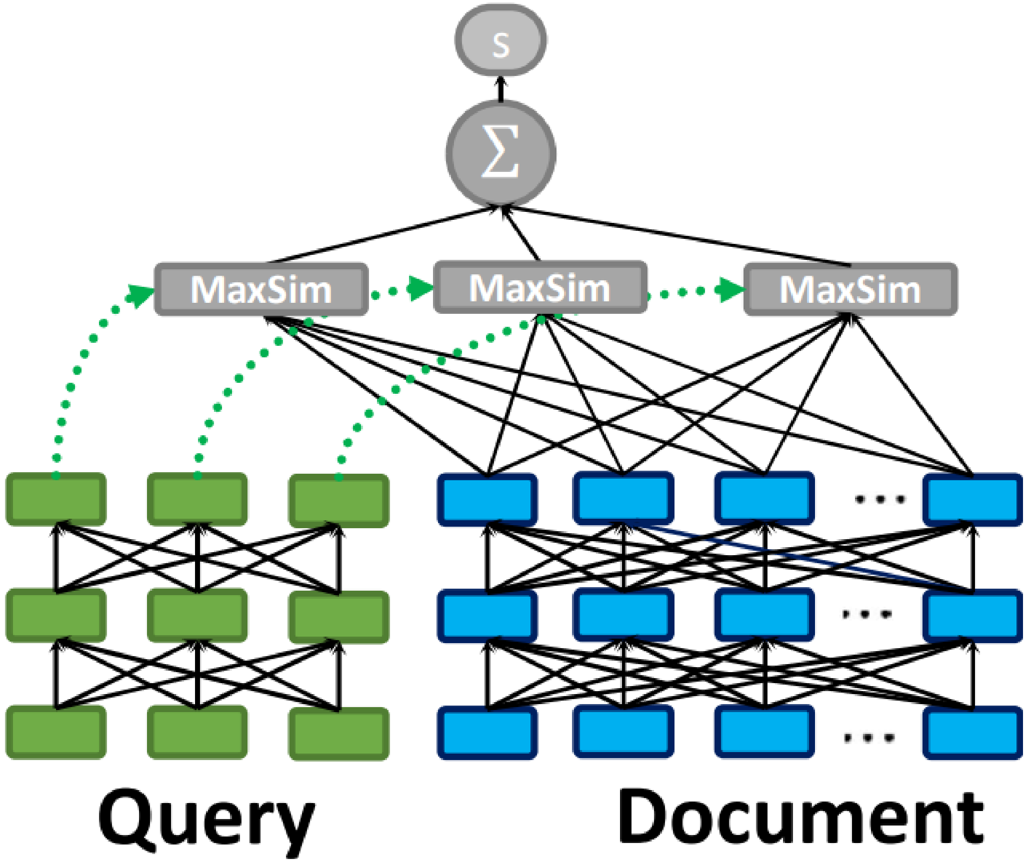
- Intuition: For each word in the query, find the most matching word in the current candidate document to compute the similarity



(d) Late Interaction

Late Interaction

- Example
  - Query representation
    $[1,0],\ [0,1]$
  - Document representation
    $[1,1],\ [0.5,0.5]$
  - For $[1,0]$, which document vector is the most similar?
  - For $[0,1]$, which document vector is the most similar?
  - $\mathrm{score}(q,d)$
    $= [1,0]\begin{bmatrix}1\\1\end{bmatrix} + [0,1]\begin{bmatrix}1\\1\end{bmatrix} = 2$



(d) Late Interaction

Late Interaction: Real Example of Matching

<u>when</u> did the transformers cartoon series come out?

[…] the animated […] The Transformers […] […] It was released […] **<u>on</u>** August 8, 1986

when did the <u>transformers</u> cartoon series come out?

[…] the animated […] The **<u>Transformers</u>** […] […] It was released […] on August 8, 1986

when did the transformers <u>cartoon</u> series come out?

[…] the **<u>animated</u>** […] The Transformers […] […] It was released […] on August 8, 1986

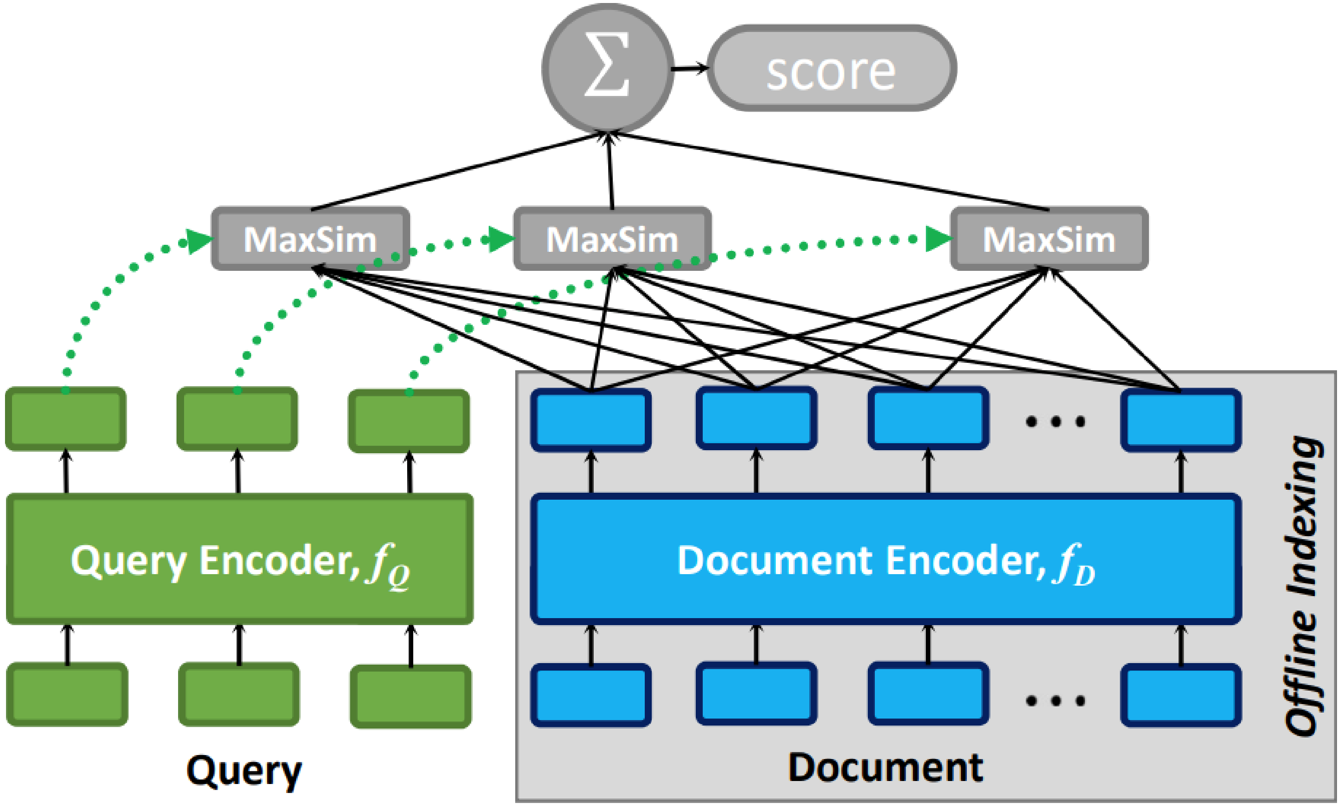
when did the transformers cartoon series <u>come</u> <u>out</u>?

[…] the animated […] The Transformers […] […] It was **<u>released</u>** […] on August 8, 1986

Offline Indexing

- For Bi-Encoder, index one vector for each candidate document

- For ColBERT, index a sequence of vectors for each candidate document



Performance of ColBERT: Reranking

- On par with BERT-Base (Cross-Encoder)
- Slightly worse than BERT-Large (Cross-Encoder)
- BUT much faster!

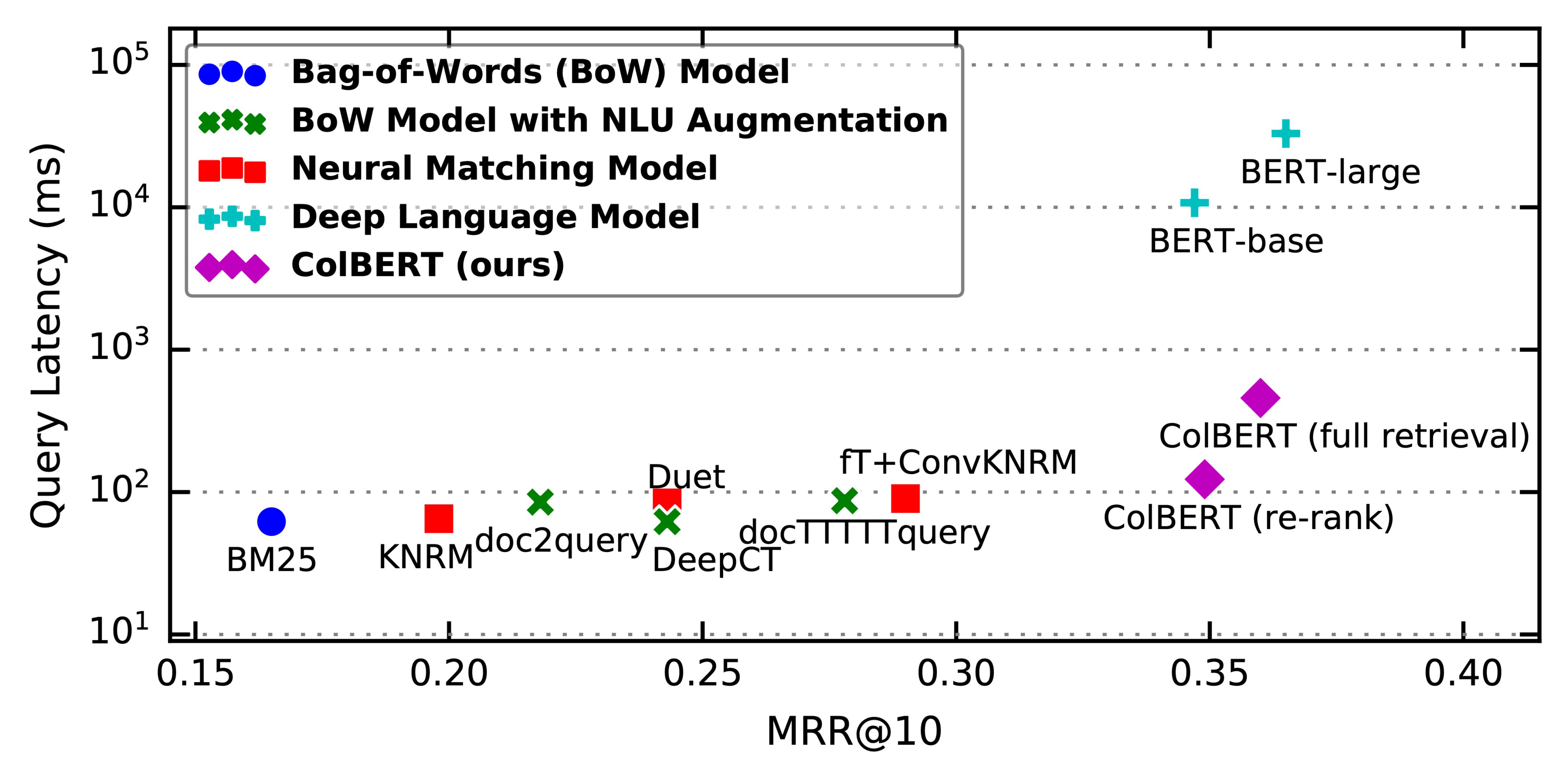
Method	MRR@10 (Dev)	MRR@10 (Eval)	Re-ranking Latency (ms)	FLOPs/query
BM25 (official)	16.7	16.5	-	-
KNRM	19.8	19.8	3	592M (0.085×)
Duet	24.3	24.5	22	159B (23×)
fastText+ConvKNRM	29.0	27.7	28	78B (11×)
$\text{BERT}_{\text{base}}$ [25]	34.7	-	10,700	97T (13,900×)
$\text{BERT}_{\text{base}}$ (our training)	36.0	-	10,700	97T (13,900×)
$\text{BERT}_{\text{large}}$ [25]	36.5	35.9	32,900	340T (48,600×)
ColBERT (over $\text{BERT}_{\text{base}}$)	34.9	34.9	61	7B (1×)

Performance of ColBERT: End-to-End Retrieval

- <10× slower than BM25 and Bi-Encoder ranking models
- BUT much more effective!

Method	**MRR@10 (Dev)**	**MRR@10 (Local Eval)**	**Latency (ms)**	**Recall@50**	**Recall@200**	**Recall@1000**
BM25 (official)	16.7	-	-	-	-	81.4
BM25 (Anserini)	18.7	19.5	62	59.2	73.8	85.7
doc2query	21.5	22.8	85	64.4	77.9	89.1
DeepCT	24.3	-	62 *(est.)*	69 [2]	82 [2]	91 [2]
docTTTTTquery	27.7	28.4	87	75.6	86.9	94.7
$\text{ColBERT}_{\text{L2}}$ (re-rank)	34.8	36.4	-	75.3	80.5	81.4
$\text{ColBERT}_{\text{L2}}$ (end-to-end)	36.0	36.7	458	82.9	92.3	96.8

Striking a Good Balance Between Bi-Encoder and Cross-Encoder



Robustness: Out-of-Domain Quality

- So far, we have looked at in-domain effectiveness evaluations
  - We had training and evaluation data for MS MARCO
  - We often want to use retrieval in new, out-of-domain settings with NO training data and NO validation data
  - This is sometimes called a "zero-shot" setting; it emphasizes generalization
  - BEIR is a popular benchmark for IR models in "zero-shot" scenarios

https://github.com/beir-cellar/beir

BEIR

Split (→)					Train	Dev	Test			Avg. Word Lengths	
Task (↓)	**Domain (↓)**	**Dataset (↓)**	**Title**	**Relevancy**	**#Pairs**	**#Query**	**#Query**	**#Corpus**	**Avg. D / Q**	**Query**	**Document**
Passage-Retrieval	Misc.	MS MARCO [45]	✗	Binary	532,761	—	6,980	8,841,823	1.1	5.96	55.98
Bio-Medical Information Retrieval (IR)	Bio-Medical	TREC-COVID [65]	✓	3-level	—	—	50	171,332	493.5	10.60	160.77
	Bio-Medical	NFCorpus [7]	✓	3-level	110,575	324	323	3,633	38.2	3.30	232.26
	Bio-Medical	BioASQ [61]	✓	Binary	32,916	—	500	14,914,602	4.7	8.05	202.61
Question Answering (QA)	Wikipedia	NQ [34]	✓	Binary	132,803	—	3,452	2,681,468	1.2	9.16	78.88
	Wikipedia	HotpotQA [76]	✓	Binary	170,000	5,447	7,405	5,233,329	2.0	17.61	46.30
	Finance	FiQA-2018 [44]	✗	Binary	14,166	500	648	57,638	2.6	10.77	132.32
Tweet-Retrieval	Twitter	Signal-1M (RT) [59]	✗	3-level	—	—	97	2,866,316	19.6	9.30	13.93
News Retrieval	News	TREC-NEWS [58]	✓	5-level	—	—	57	594,977	19.6	11.14	634.79
	News	Robust04 [64]	✗	3-level	—	—	249	528,155	69.9	15.27	466.40
Argument Retrieval	Misc.	ArguAna [67]	✓	Binary	—	—	1,406	8,674	1.0	192.98	166.80
	Misc.	Touché-2020 [6]	✓	3-level	—	—	49	382,545	19.0	6.55	292.37
Duplicate-Question Retrieval	StackEx.	CQADupStack [25]	✓	Binary	—	—	13,145	457,199	1.4	8.59	129.09
	Quora	Quora	✗	Binary	—	5,000	10,000	522,931	1.6	9.53	11.44
Entity-Retrieval	Wikipedia	DBPedia [21]	✓	3-level	—	67	400	4,635,922	38.2	5.39	49.68
Citation-Prediction	Scientific	SCIDOCS [9]	✓	Binary	—	—	1,000	25,657	4.9	9.38	176.19
Fact Checking	Wikipedia	FEVER [60]	✓	Binary	140,085	6,666	6,666	5,416,568	1.2	8.13	84.76
	Wikipedia	Climate-FEVER [14]	✓	Binary	—	—	1,535	5,416,593	3.0	20.13	84.76
	Scientific	SciFact [68]	✓	Binary	920	—	300	5,183	1.1	12.37	213.63

Robustness: Out-of-Domain NDCG@10

IR Task	Classical IR **BM25**	Interaction Models **ELECTRA re-ranker**	Representation Similarity **DPR**	Representation Similarity **SBERT**	Late Interaction **ColBERT**
BioMed	48	**49**	22	34	**49**
QA	38	**51**	33	41	48
Tweet	**39**	31	16	26	27
News	37	**43**	16	37	39
Arguments	**52**	35	15	34	25
Duplicates	53	56	20	58	**60**
Entity	29	38	26	34	**39**
Citation	**16**	15	8	13	15
Fact-Check	48	52	34	47	**54**
Overall Avg	42	**45**	23	39	44

Robustness: Out-of-Domain Recall@100

IR Task	Classical IR BM25	Interaction Models ELECTRA re-ranker	Representation Similarity DPR	Representation Similarity SBERT	Late Interaction ColBERT
BioMed	**45**	**45**	23	35	**45**
QA	67	67	60	68	**75**
Tweet	**38**	**38**	16	26	28
News	**40**	**40**	22	37	37
Arguments	**70**	**70**	46	62	61
Duplicates	77	77	44	79	**81**
Entity	38	38	35	40	**46**
Citation	**35**	**35**	22	30	34
Fact-Check	71	71	65	74	**75**
Overall Avg	59	59	43	57	**61**

How to build effective "zero-shot" rankers?

Next Lecture

- How to leverage neural networks and embeddings in recommender systems (e.g., for collaborative filtering)?
- Quiz 3!
  - All policies are the same as Quiz 1 (number of questions, time limit, grading, etc.)
  - Scope:
    - Lecture 14 (Bayesian Personalized Ranking)
    - Lecture 15 (Word Embedding)
    - Lecture 16 (Neural Ranking)
    - Lecture 17 (Transformer, BERT)
    - Lecture 18 (BERT-Based Ranking)
    - Homework 2

Thank You!

Course Website: https://yuzhang-teaching.github.io/CSCE670-F25.html