



CSCE 670 - Information Storage and Retrieval

Week 9: BERT, BERT-Based Ranking

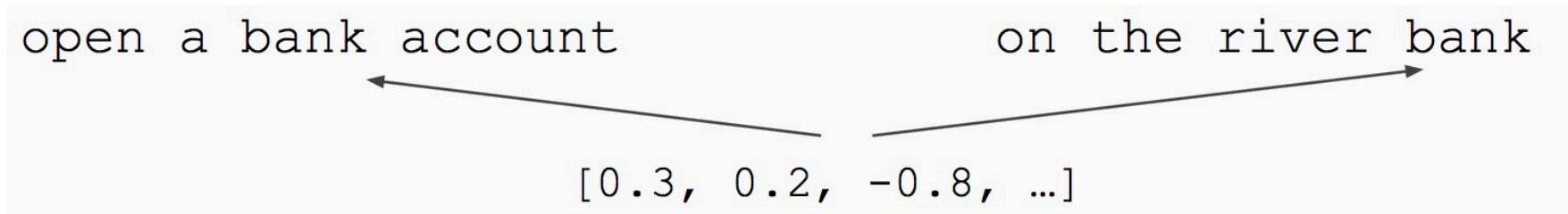
Yu Zhang

yuzhang@tamu.edu

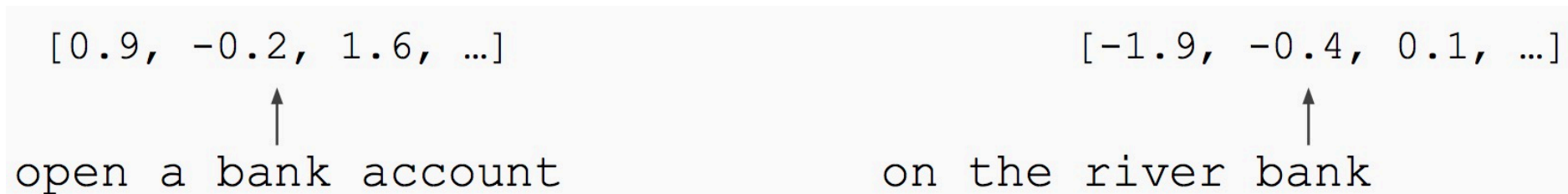
Course Website: <https://yuzhang-teaching.github.io/CSCE670-F26.html>

Recap: Limitations of word2vec-based Ranking

- Word embeddings are applied in a context-free manner!



- Is it possible to have different vectors of the same **word** given different **contexts**?



Transformer [Vaswani et al., NIPS 2017]

Attention Is All You Need

Attention is all you need

[A Vaswani](#), [N Shazeer](#), [N Parmar](#), [J Uszkoreit](#), [L Jones](#), [AN Gomez](#), [Ł Kaiser](#), [I Polosukhin](#)

Advances in neural information processing systems, 2017 • [proceedings.neurips.cc](#)

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder and decoder configuration. The best performing such models also connect the encoder and decoder through an attention mechanism. We propose a novel, simple network architecture based solely on an attention mechanism, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more

SHOW MORE ▾

☆ Save 📄 Cite Cited by 273745 Related articles All 26 versions 🔗

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

†
to
edu
Łukasz Kaiser*
Google Brain
lukaszkaizer@google.com

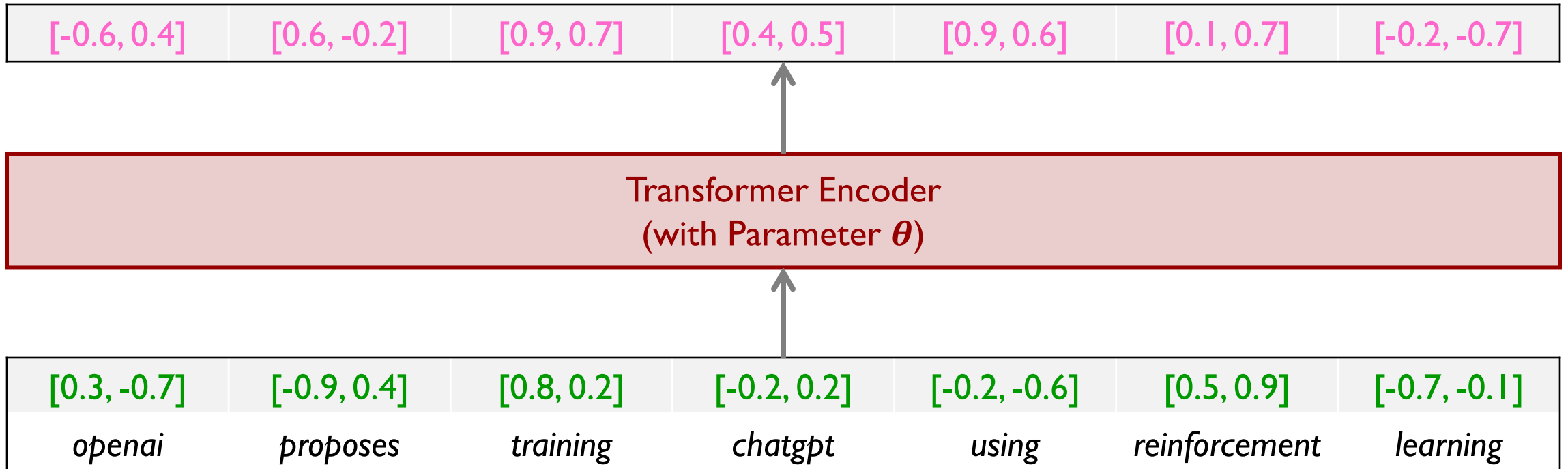
hin* ‡
@gmail.com

Transformer as a Black Box

- Transformer is a **neural network**
- It has two types of architecture: **encoder** and **decoder**
- In this lecture, we focus on the **Transformer encoder**
- 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 $e_{w_1}, e_{w_2}, \dots, e_{w_L}$
- Then, the output is a sequence of contextualized word vectors $h_{w_1}, h_{w_2}, \dots, h_{w_L}$
 - The output vector h_{w_i} captures the meaning of w_i by considering the entire input sequence as w_i 's context
 - $h_{w_i} = \text{Transformer}(e_{w_i} | e_{w_1}, e_{w_2}, \dots, e_{w_L})$

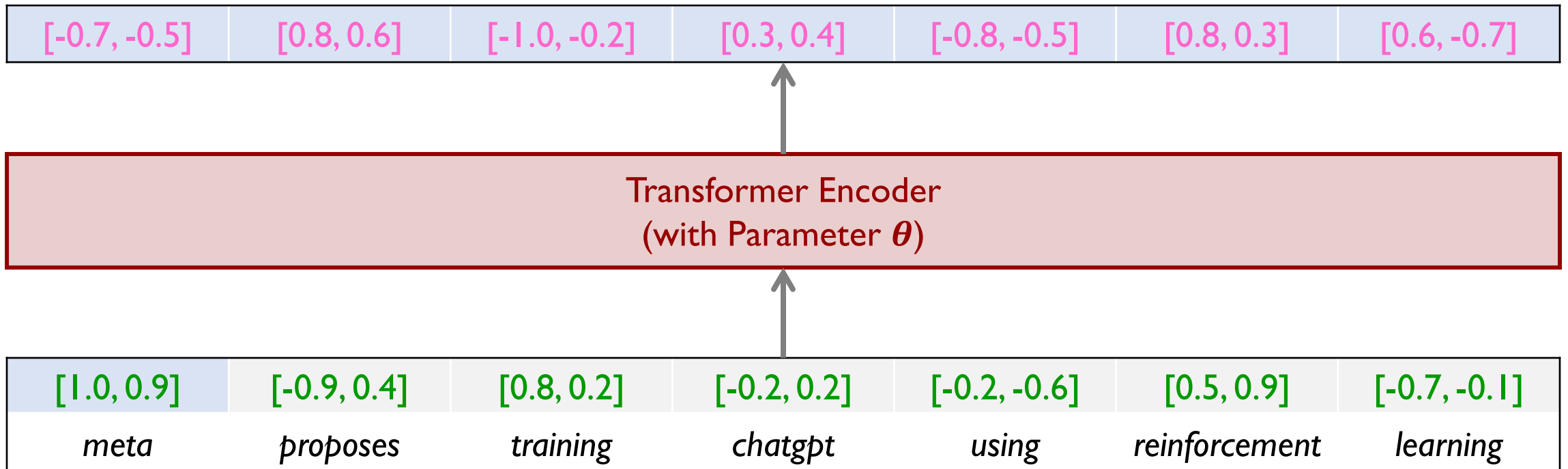
Transformer as a Black Box

- $h_{w_i} = \text{Transformer}(\mathbf{e}_{w_i} | \mathbf{e}_{w_1}, \mathbf{e}_{w_2}, \dots, \mathbf{e}_{w_L})$



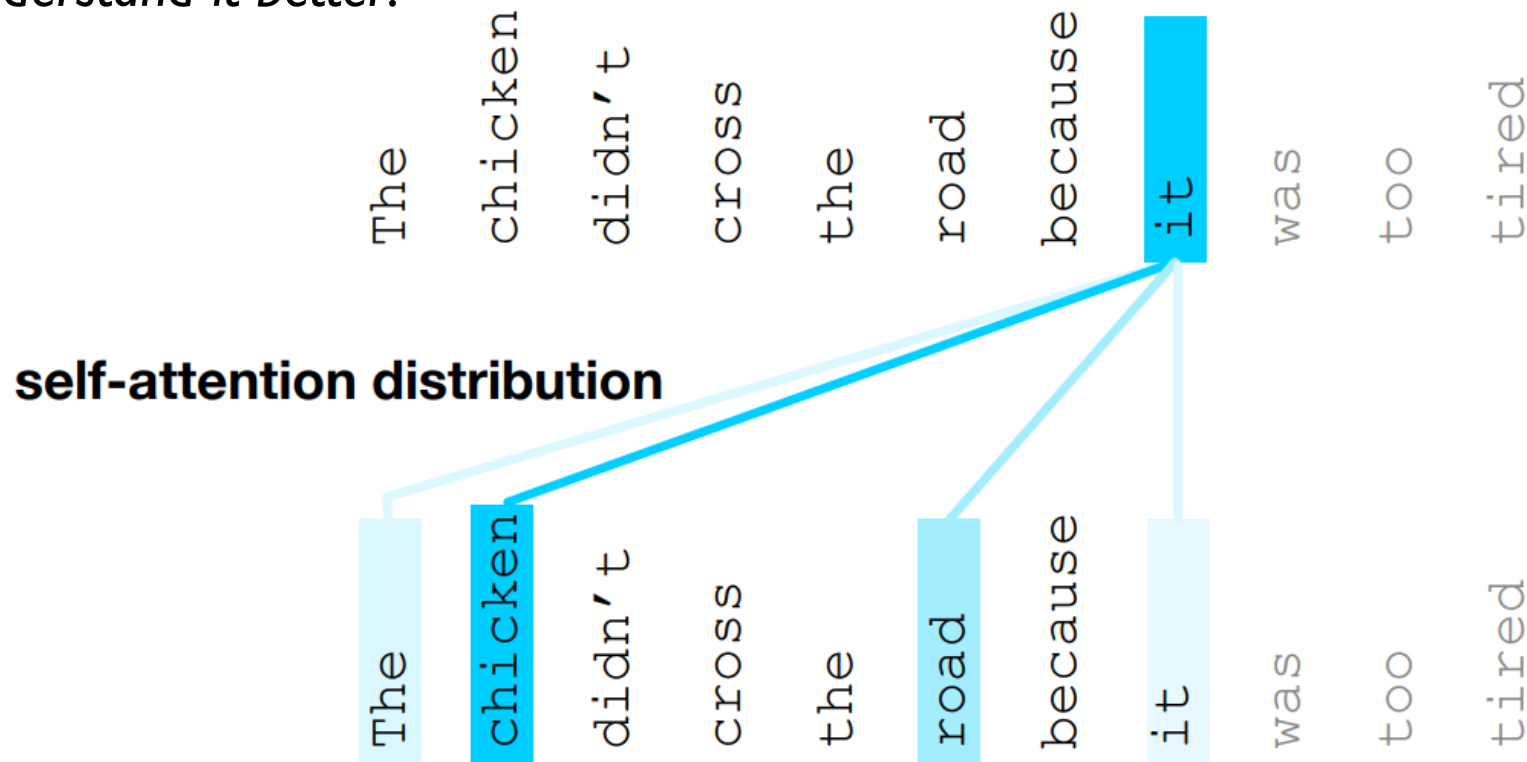
Transformer as a Black Box

- $h_{w_i} = \text{Transformer}(\mathbf{e}_{w_i} | \mathbf{e}_{w_1}, \mathbf{e}_{w_2}, \dots, \mathbf{e}_{w_L})$



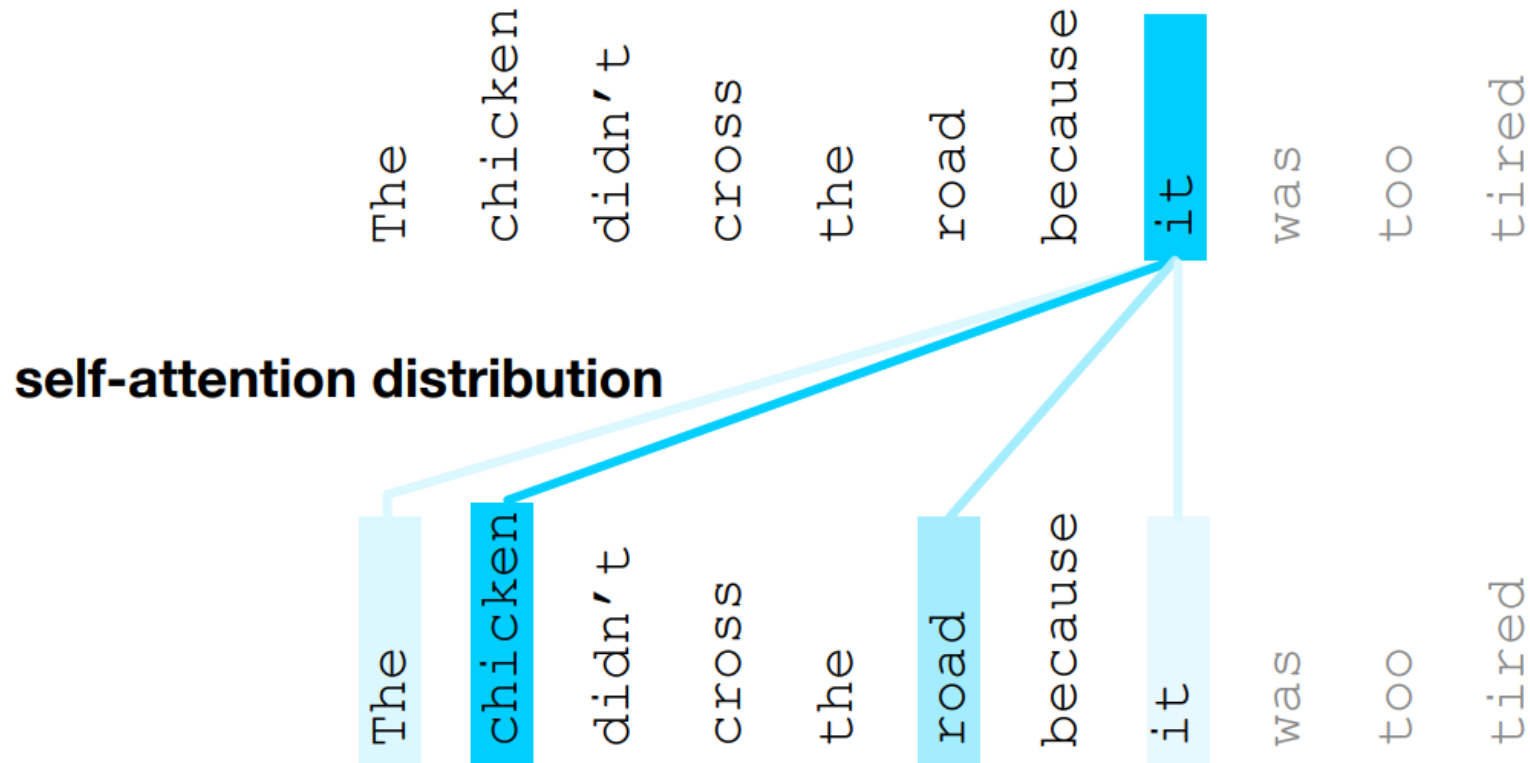
How does the model transform the input into the output?

- “**(Self-)Attention**”: weigh the importance of different words in a sequence when processing a specific word
 - *When I am looking at this word, which other words should I pay attention to in order to understand it better?*



Self-Attention: A Simplified Example

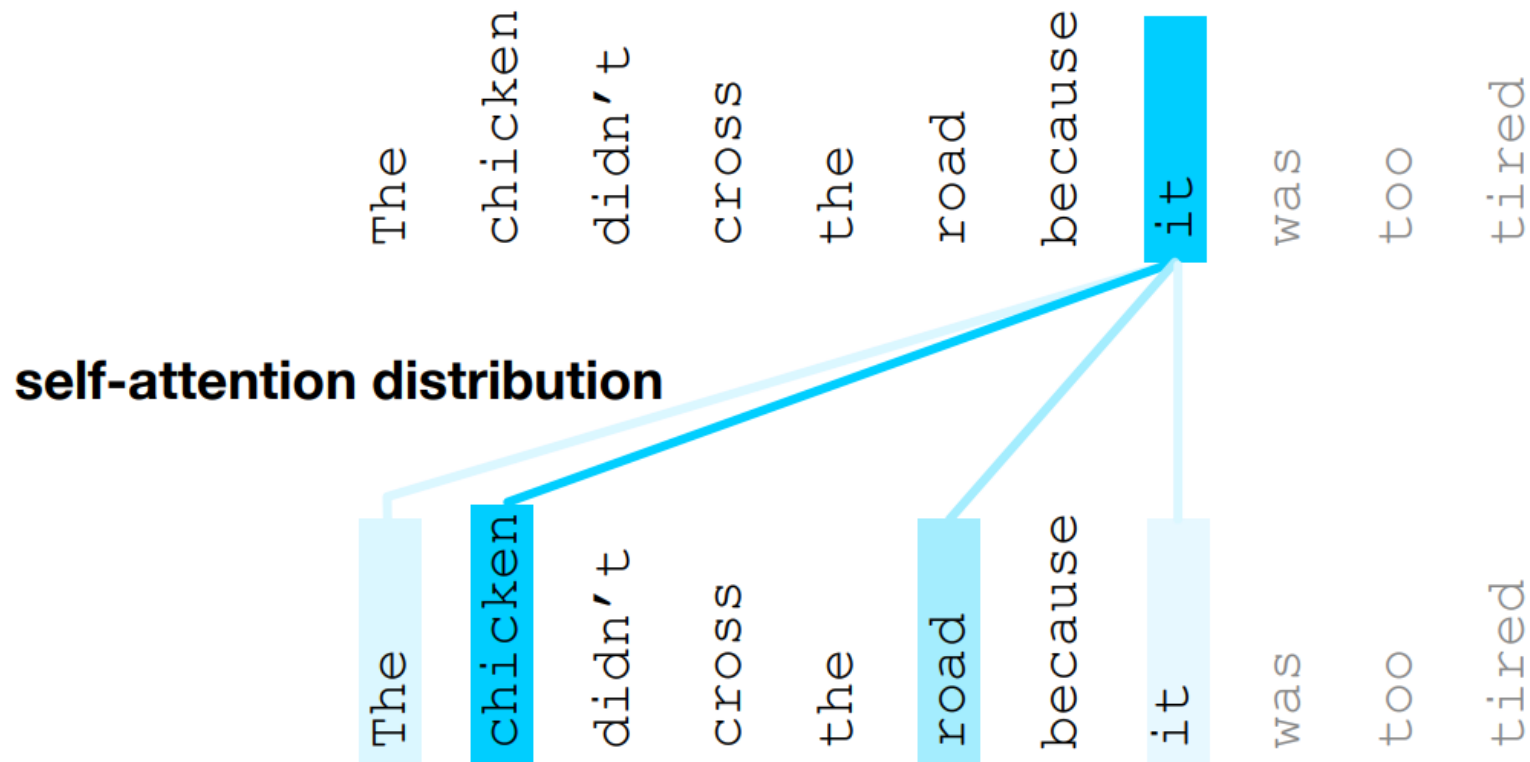
- $\mathbf{h}_{w_i} = \sum_{j=1}^L a_{ij} \mathbf{e}_{w_j}$
- a_{ij} : attention score, where $\sum_{j=1}^L a_{ij} = 1$



Self-Attention: A Simplified Example

- $h_{w_i} = \sum_{j=1}^L \text{Softmax}(\mathbf{e}_{w_i}^T \mathbf{e}_{w_j}) \cdot \mathbf{e}_{w_j}$
- $\text{Softmax}(\mathbf{e}_{w_i}^T \mathbf{e}_{w_j}) = \frac{\exp(\mathbf{e}_{w_i}^T \mathbf{e}_{w_j})}{\sum_{k=1}^L \exp(\mathbf{e}_{w_i}^T \mathbf{e}_{w_k})}$

The weight is determined by the similarity between the context word and the center word.



Self-Attention: Query, Key, Value

- Each word in self-attention is represented by three different vectors
- **Query (Q)**: Represent the current word for which information is being sought
- **Key (K)**: Represent the reference (context) used for comparison with the query
- **Value (V)**: Represent the actual content of each token, which will be aggregated into the final output

$$h_{w_i} = \sum_{j=1}^L \text{Softmax}(e_{w_i}^T e_{w_j}) \cdot e_{w_j}$$

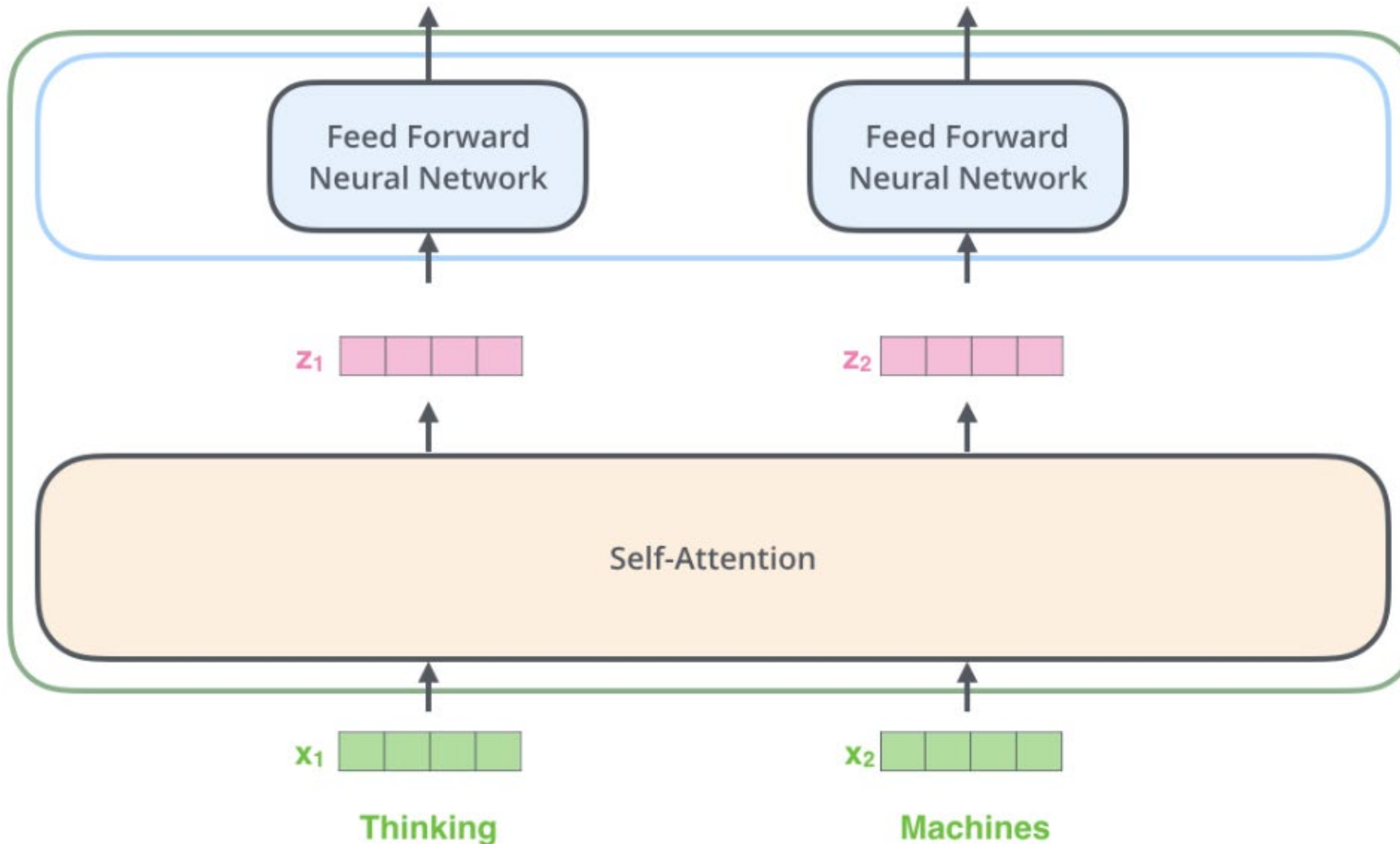
The diagram illustrates the components of the self-attention formula. Three arrows point from the terms in the equation to labels in yellow boxes below:
1. An arrow from $e_{w_i}^T$ (purple) points to a box labeled "Query" (purple).
2. An arrow from e_{w_j} (orange) in the dot product points to a box labeled "Key" (orange).
3. An arrow from e_{w_j} (blue) in the multiplication points to a box labeled "Value" (blue).

Extended Content: Transformer Architecture Details
(will not appear in quizzes or the exam)

Self-Attention in Transformer: Learning Three Matrices

- Input word vector: $\mathbf{e}_{w_i}$
- Learn a query matrix $\mathbf{W}^Q$: $\mathbf{q}_i = \mathbf{e}_{w_i} \mathbf{W}^Q$
- Learn a key matrix $\mathbf{W}^K$: $\mathbf{k}_i = \mathbf{e}_{w_i} \mathbf{W}^K$
- Learn a value matrix $\mathbf{W}^V$: $\mathbf{v}_i = \mathbf{e}_{w_i} \mathbf{W}^V$
- Compute attention scores with query and key: $a_{ij} = \text{Softmax}\left(\frac{\mathbf{q}_i^T \mathbf{k}_j}{\sqrt{d}}\right)$
 - The dot product of two vectors usually has an expected magnitude proportional to $\sqrt{d}$, where d is the dimensionality of $\mathbf{q}_i$ and $\mathbf{k}_j$
 - Divide the attention score by $\sqrt{d}$ to avoid extremely large values in $\text{Softmax}(\cdot)$
- Sum the value vectors weighted by attention scores: $\mathbf{z}_i = \sum_{j=1}^L a_{ij} \mathbf{v}_j$

Self-Attention in Transformer: Visualization

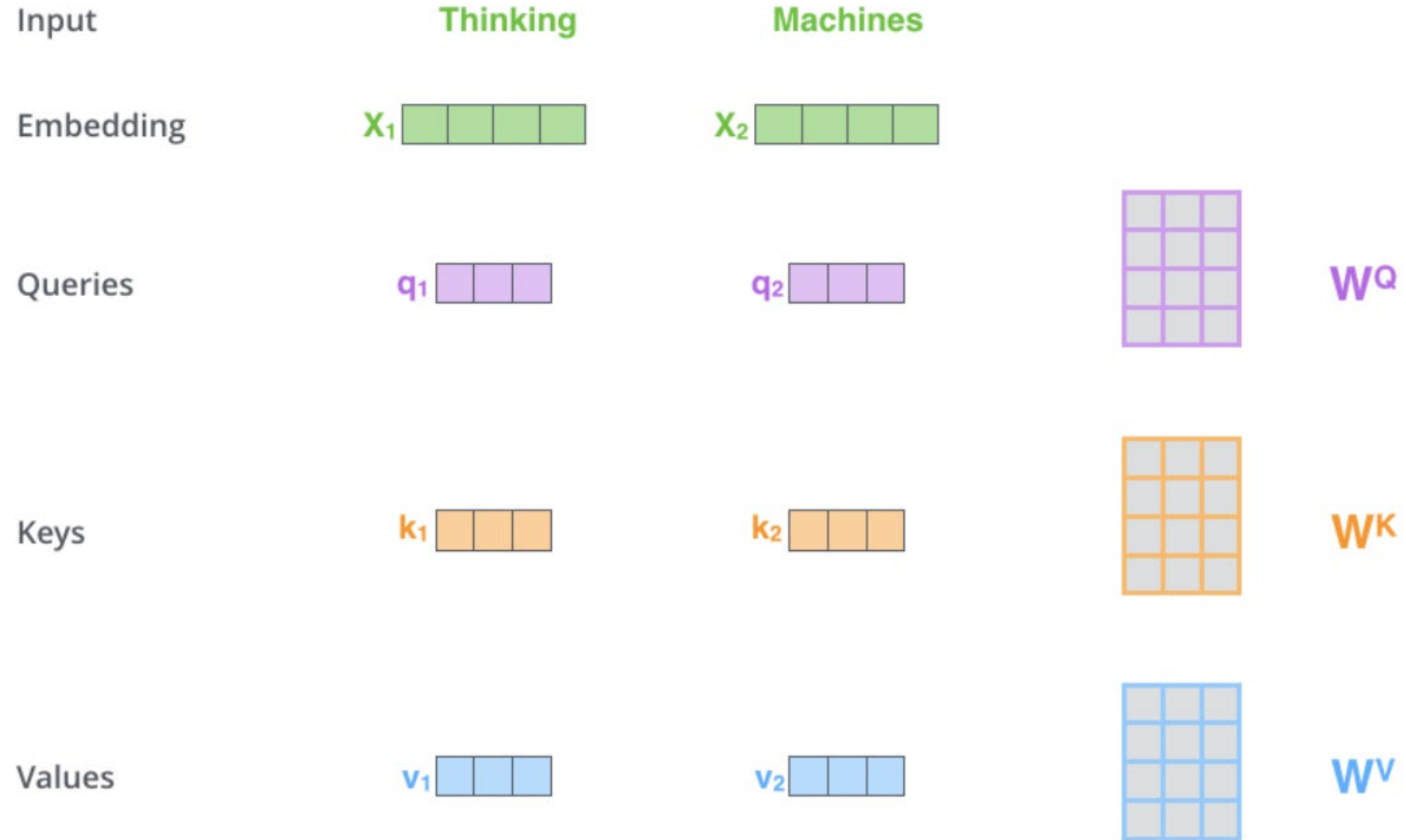


$h_{w_i} = \text{FFN}(z_i)$:
output of one
Transformer layer

z_i : output of
self-attention

x_i : input vector
(i.e., e_{w_i})

Self-Attention in Transformer: Visualization

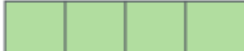


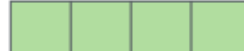
Input

Thinking

Machines

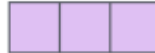
Embedding

x_1 

x_2 

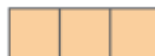
Queries

q_1 

q_2 

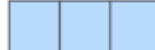
Keys

k_1 

k_2 

Values

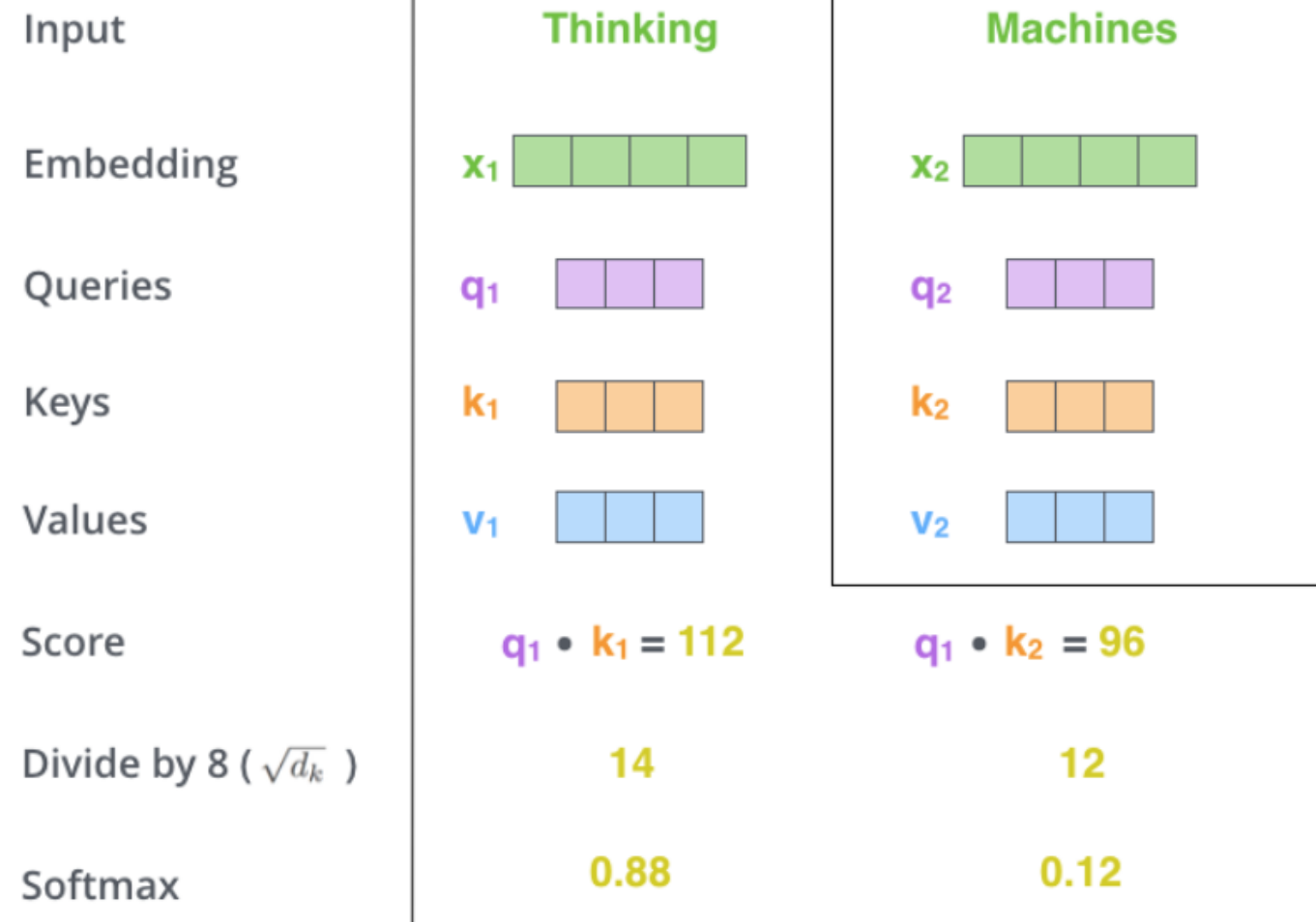
v_1 

v_2 

Score

$$q_1 \cdot k_1 = 112$$

$$q_1 \cdot k_2 = 96$$



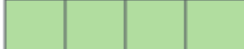
Let's assume these q_i and k_j vectors are 64-dimensional

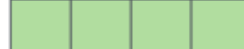
Input

Thinking

Machines

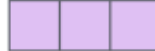
Embedding

x_1 

x_2 

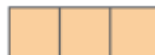
Queries

q_1 

q_2 

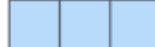
Keys

k_1 

k_2 

Values

v_1 

v_2 

Score

$q_1 \cdot k_1 = 112$

$q_1 \cdot k_2 = 96$

Divide by 8 ($\sqrt{d_k}$)

14

12

Softmax

0.88

0.12

Softmax

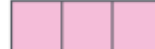
X

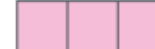
Value

v_1 

v_2 

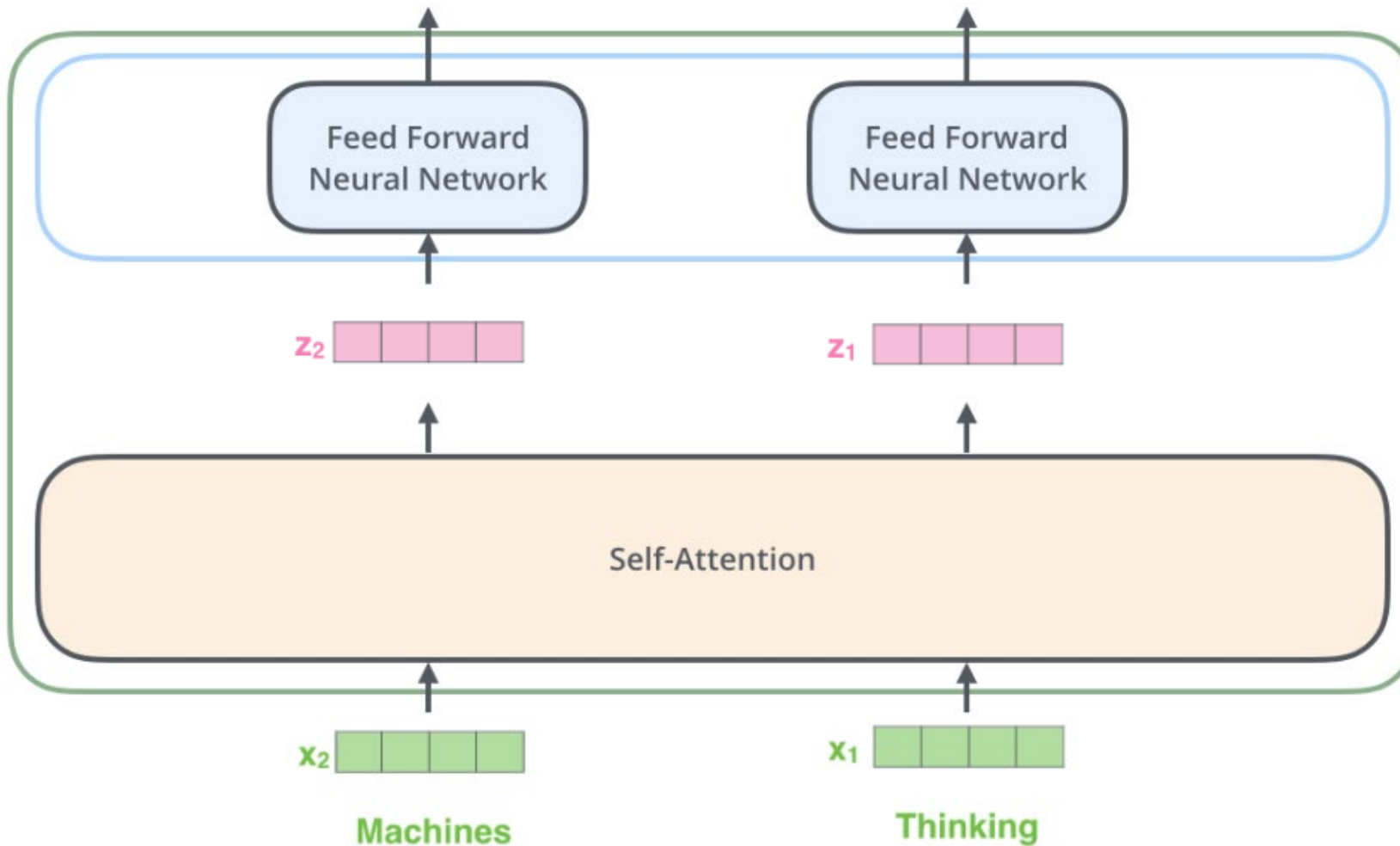
Sum

z_1 

z_2 

Position Encoding

- What will happen to the output if I swap “*Thinking*” and “*Machines*”?



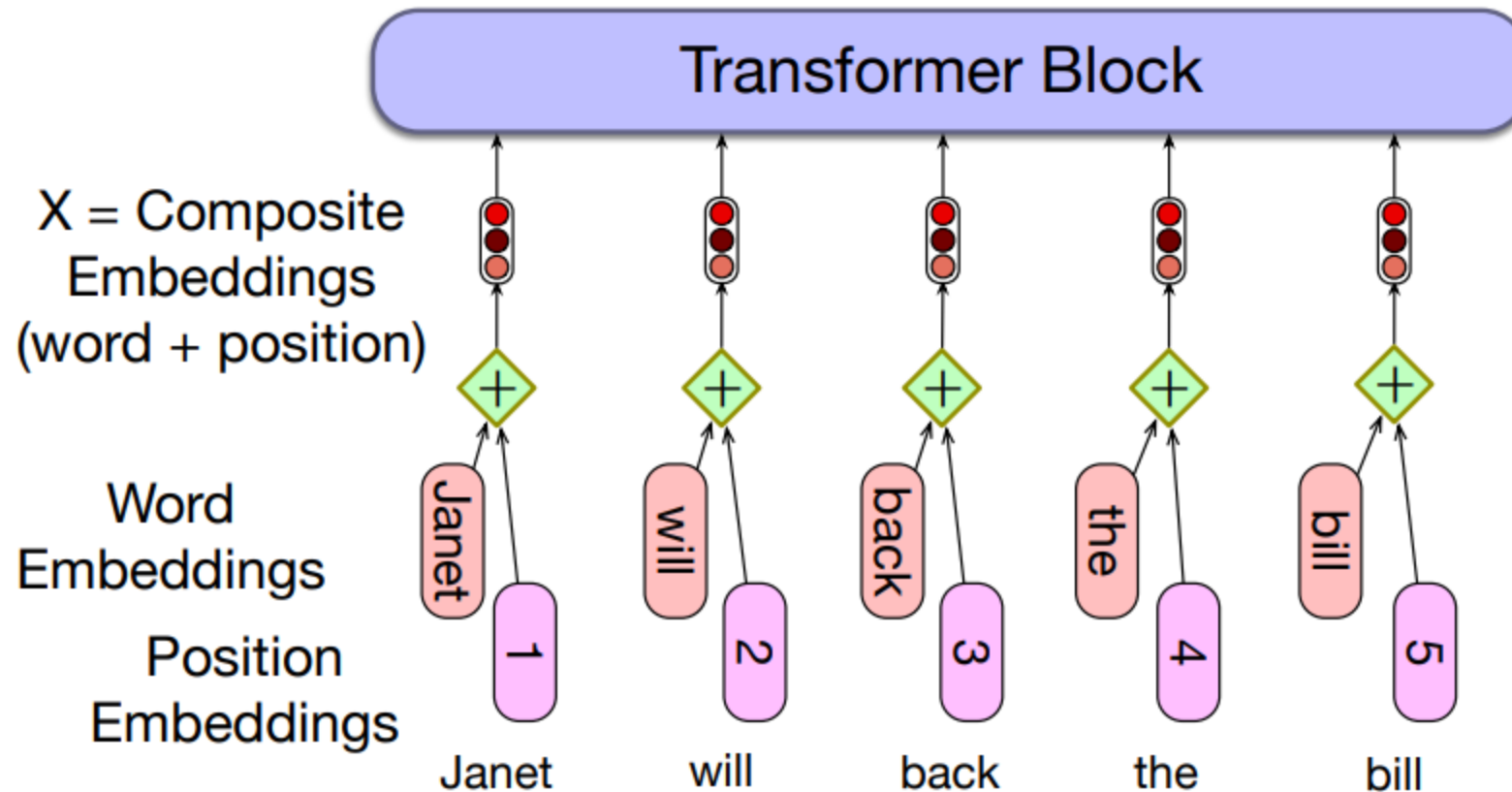
Position Encoding

- What if there are two sentences?
 - “*a dog bites a man*”
 - “*a man bites a dog*”
- The order does not affect the output vector of each word!
- But language is **order-sensitive**!
- **Solution**: Add a learnable position encoding vector p_i

input	$e_a + p_1$	$e_{\text{dog}} + p_2$	$e_{\text{bites}} + p_3$	$e_a + p_4$	$e_{\text{man}} + p_5$
word	<i>a</i>	<i>dog</i>	<i>bites</i>	<i>a</i>	<i>man</i>

input	$e_a + p_1$	$e_{\text{man}} + p_2$	$e_{\text{bites}} + p_3$	$e_a + p_4$	$e_{\text{dog}} + p_5$
word	<i>a</i>	<i>man</i>	<i>bites</i>	<i>a</i>	<i>dog</i>

Position Encoding: Visualization



Tokenization: Handling Out-of-Vocabulary Words

- Byte-Pair Encoding (BPE)
- **Intuition**: start with a character-level vocabulary and iteratively merge the most frequent pairs of tokens
- **Step 1 (Initialization)**: let vocabulary be the set of all individual characters: $\{A, B, C, D, \dots, a, b, c, d, \dots\}$
- **Step 2 (Frequency counting)**: count all adjacent symbol pairs (could be a single character or a previously merged pair) in the training corpus
- **Step 3 (Pair merging)**: merge the most frequent pair of symbols (e.g., “*t*”, “*h*” $\rightarrow$ “*th*”)
- **Step 4 (Update corpus)**: replace all instances of the merged pair in the corpus with the new token & update the frequency of pairs
- **Step 5 (Repeat)**: repeat the process of counting, merging, and updating until a predefined number of merges (or vocabulary size) is reached

Example

- Suppose we have the following corpus:
 - “set new new renew reset renew”

Special “begin-of-word” character
(distinguish between subword
units vs. whole word)

corpus

2	␣	n	e	w		
2	␣	r	e	n	e	w
1	s	e	t			
1	␣	r	e	s	e	t

vocabulary

␣, e, n, r, s, t, w

Example

- Suppose we have the following corpus:
 - “set new new renew reset renew”

corpus

```
2      _ n e w
2      _ r e n e w
1      s e t
1      _ r e s e t
```

merging
“n” and “e”



corpus

```
2      _ ne w
2      _ r e ne w
1      s e t
1      _ r e s e t
```

vocabulary

_ , e , n , r , s , t , w

vocabulary

_ , e , n , r , s , t , w , ne

Example

- Suppose we have the following corpus:
 - “set new new renew reset renew”

corpus

```
2  _ ne w
2  _ r e ne w
1  s e t
1  _ r e s e t
```

merging “ne”
and “w”



corpus

```
2  _ new
2  _ r e new
1  s e t
1  _ r e s e t
```

vocabulary

_ , e , n , r , s , t , w , ne

vocabulary

_ , e , n , r , s , t , w , ne ,
new

Example

- Suppose we have the following corpus:
 - “set new new renew reset renew”

corpus

2 _ new
2 _ r e new
1 s e t
1 _ r e s e t



corpus

2 _ new
2 _re new
1 s e t
1 _re s e t

vocabulary

_, e, n, r, s, t, w, ne,
new

vocabulary

_, e, n, r, s, t, w, ne,
new, _r, _re

Example

- Suppose we have the following corpus:
 - “*set new new renew reset renew*”
- If we continue, the next merges are:

merge	current vocabulary
(<code>_</code> , new)	<code>_</code> , e, n, r, s, t, w, ne, new, <code>_r</code> , <code>_re</code> , <code>_new</code>
(<code>_re</code> , new)	<code>_</code> , e, n, r, s, t, w, ne, new, <code>_r</code> , <code>_re</code> , <code>_new</code> , <code>_renew</code>
(s, e)	<code>_</code> , e, n, r, s, t, w, ne, new, <code>_r</code> , <code>_re</code> , <code>_new</code> , <code>_renew</code> , se
(se, t)	<code>_</code> , e, n, r, s, t, w, ne, new, <code>_r</code> , <code>_re</code> , <code>_new</code> , <code>_renew</code> , se, set

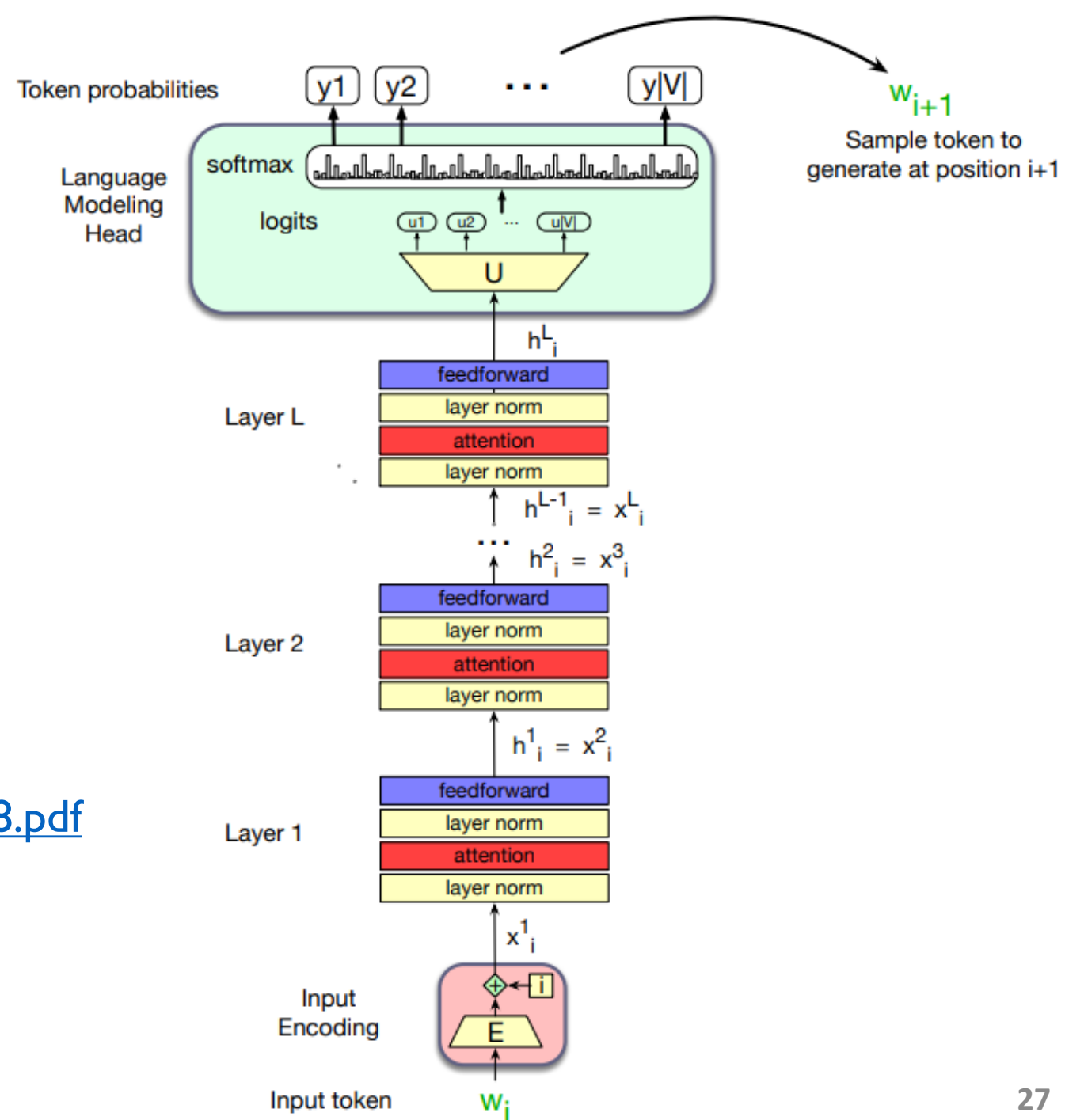
- Given a new sentence, how to tokenize it?
 - Just run (greedily based on training data frequency) on the merge rules we have learned from the training data on the new sentence

Still many missing details in the Transformer!

- Multi-head attention
- Feedforward network
- Layer normalization
- Residual connection

- Refer to:

<https://web.stanford.edu/~jurafsky/slp3/8.pdf>



Training Transformers with Self-Supervision

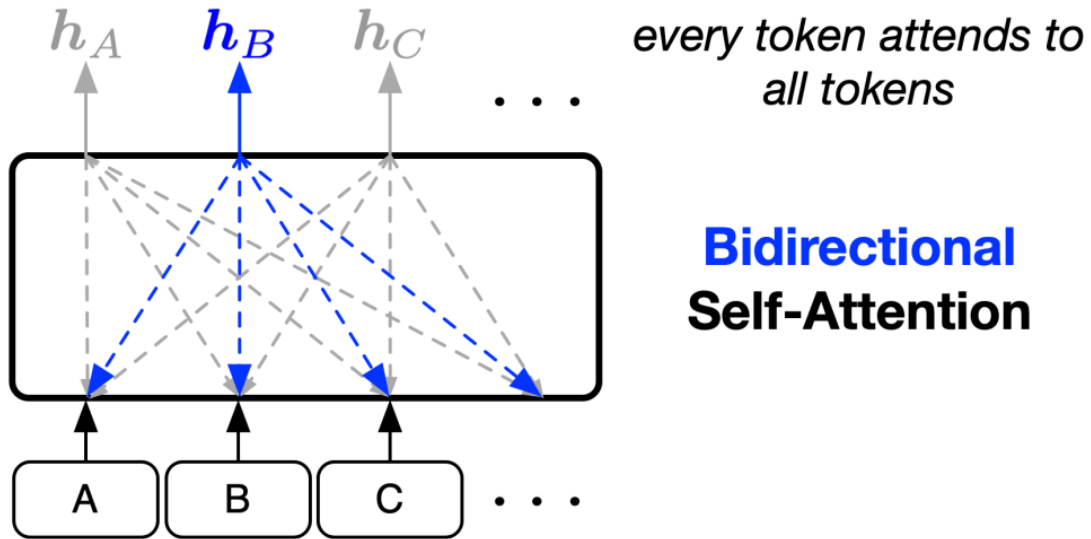
Information in Raw Texts

- *Verb:* the task is to all the structured chemical reactions from papers
- *Preposition:* a liquid handler equipped two microplates
- *Time:* the Transformer paper was published in
- *Location:* data from the University of MD Anderson Cancer Center
- *Math:* the sequence goes 1, 1, 2, 3, 5, 8, 13, 21,
- *Chemistry:* sugar is composed of carbon, hydrogen, and
- ...
- How to harvest underlying patterns, structures, and semantic knowledge from raw texts?
 - Train the model to predict masked tokens given their contexts

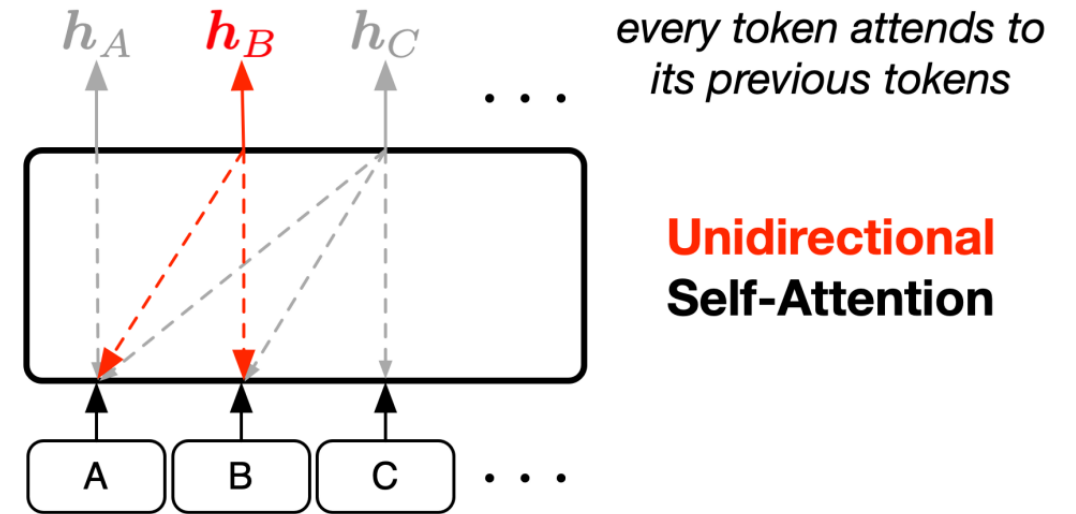
Information in Raw Texts

- *Verb*: the task is to extract all the structured chemical reactions from papers
- *Preposition*: a liquid handler equipped with two microplates
- *Time*: the Transformer paper was published in 2017
- *Location*: data from the University of Texas MD Anderson Cancer Center
- *Math*: the sequence goes 1, 1, 2, 3, 5, 8, 13, 21, 34
- *Chemistry*: sugar is composed of carbon, hydrogen, and oxygen
- ...
- Encoder-based language models (e.g., **BERT**) – predict a token from **all other tokens** in the input sequence
chatgpt learns from vast amounts of text data and generates responses ...
- Decoder-based language models (e.g., **ChatGPT**) – predict a token from **all previous tokens**
chatgpt learns from vast amounts of text data and generates responses ...

Two Types of Transformer Architecture



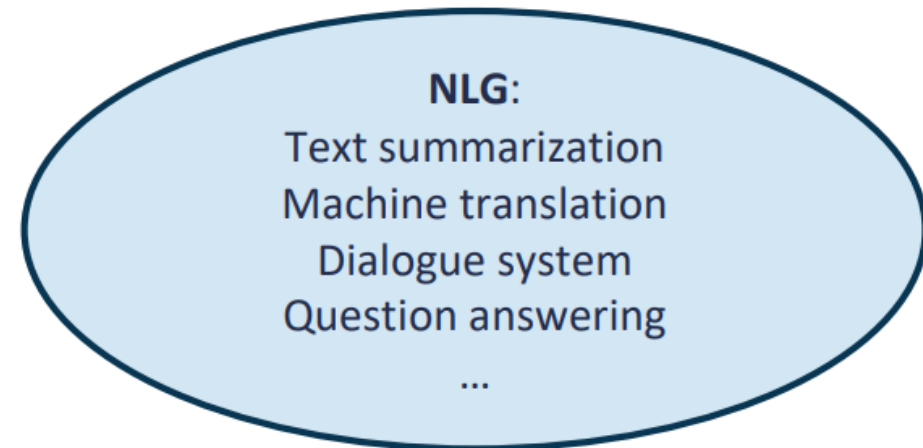
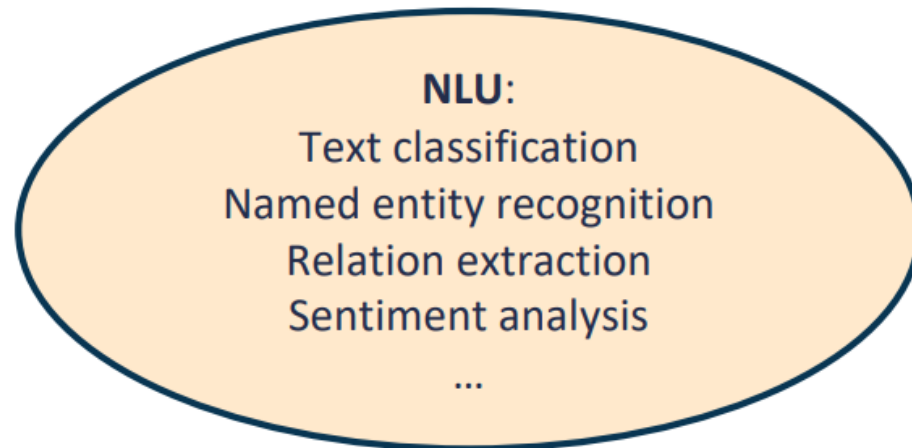
$q1 \cdot k1$	$q1 \cdot k2$	$q1 \cdot k3$	$q1 \cdot k4$
$q2 \cdot k1$	$q2 \cdot k2$	$q2 \cdot k3$	$q2 \cdot k4$
$q3 \cdot k1$	$q3 \cdot k2$	$q3 \cdot k3$	$q3 \cdot k4$
$q4 \cdot k1$	$q4 \cdot k2$	$q4 \cdot k3$	$q4 \cdot k4$



$q1 \cdot k1$	$-\infty$	$-\infty$	$-\infty$
$q2 \cdot k1$	$q2 \cdot k2$	$-\infty$	$-\infty$
$q3 \cdot k1$	$q3 \cdot k2$	$q3 \cdot k3$	$-\infty$
$q4 \cdot k1$	$q4 \cdot k2$	$q4 \cdot k3$	$q4 \cdot k4$

Encoder vs. Decoder

- Encoder:
 - Each token can attend to all other tokens **to better learn its representation vector**
 - Suitable for natural language understanding (NLU) tasks
- Decoder:
 - Each token can only attend to previous tokens **to predict the next token**
 - Suitable for natural language generation (NLG) tasks



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

Bert: Pre-training of deep bidirectional transformers for language understanding

J Devlin, [MW Chang](#), [K Lee](#)... - Proceedings of the 2019 ..., 2019 - aclanthology.org

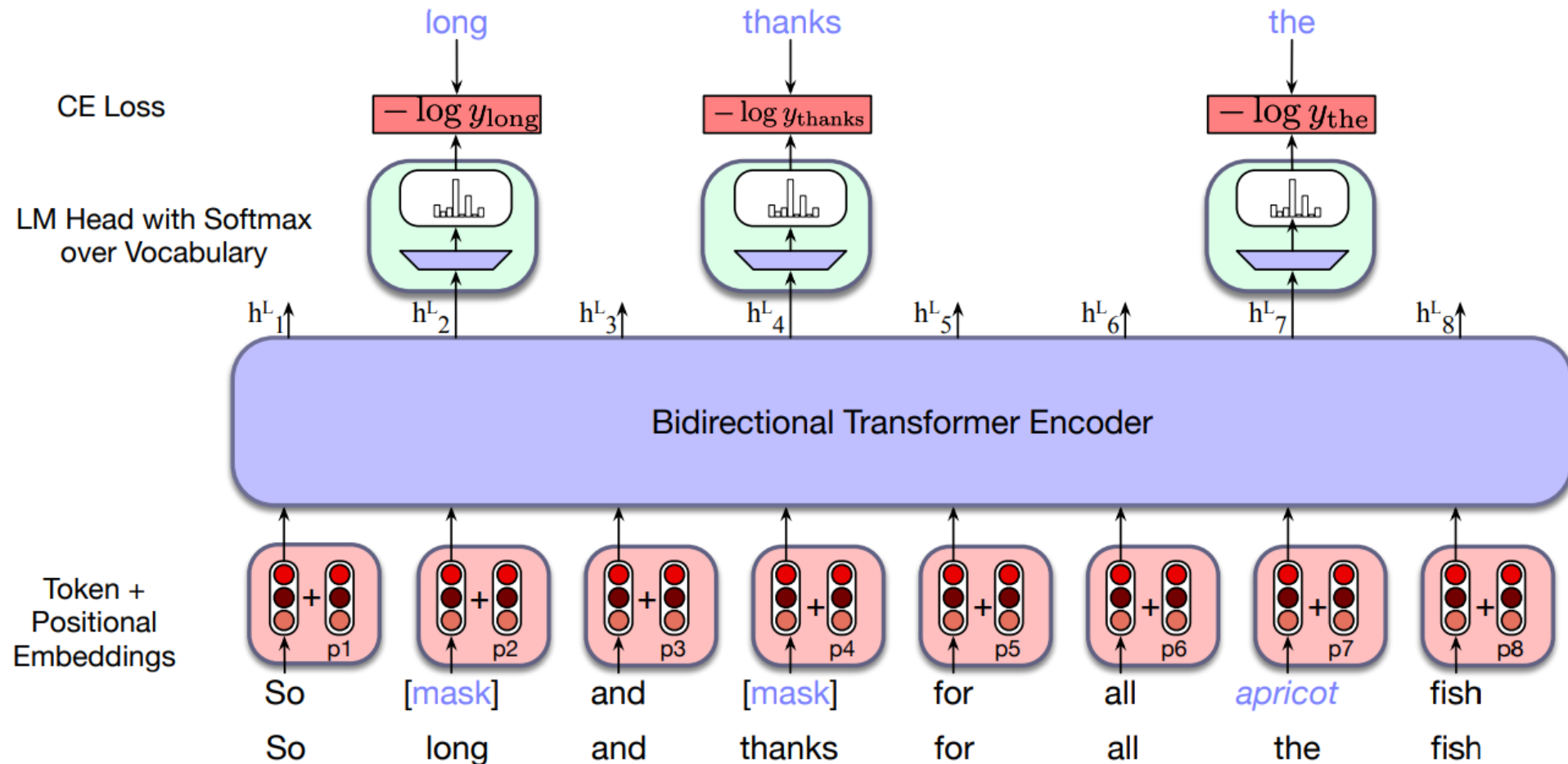
We introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of ...

☆ Save  Cite Cited by 184652 Related articles All 56 versions 

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that treat the pre-trained representations as additional features. The fine-tuning approach, such as Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal

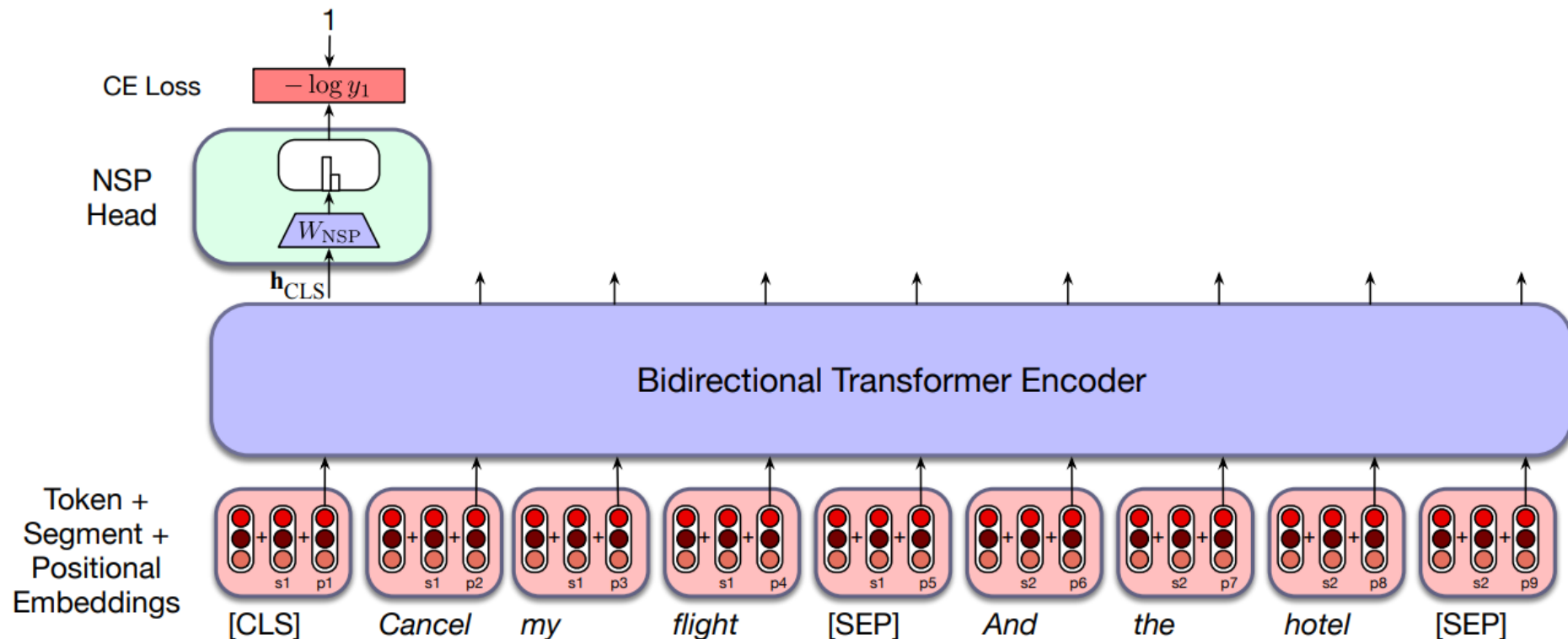
BERT Pre-training

- **Task I – 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.



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) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
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
BERT _{BASE}	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{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]

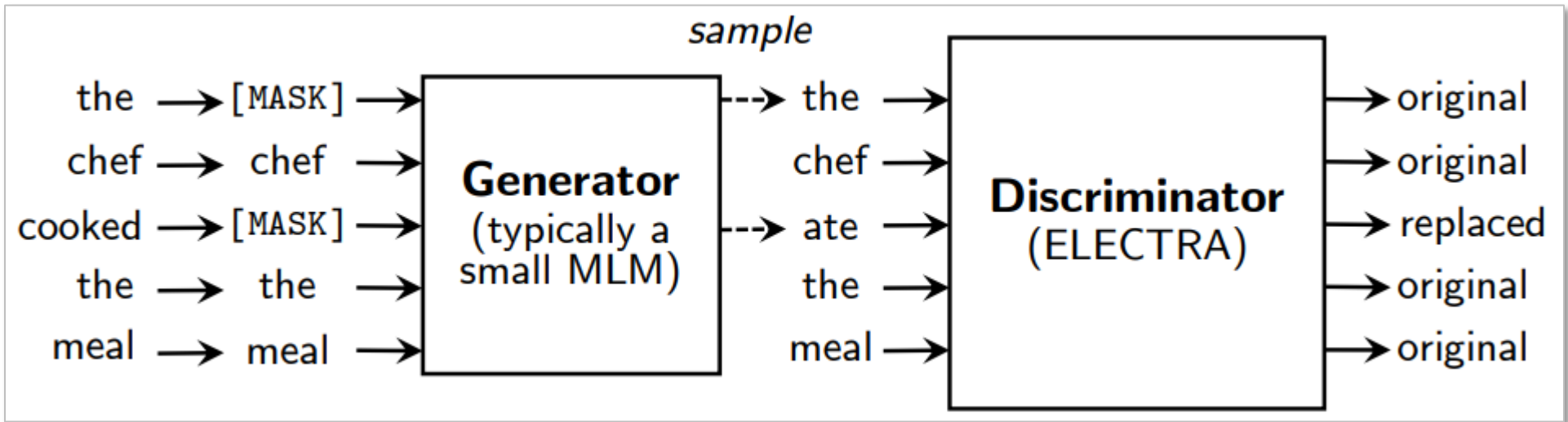
- Next Sentence Prediction (NSP) is not helpful!
- Only use Masked Language Modeling (MLM)
- Pretrain on longer sequences
- Pretrain the model for longer, with bigger batches
- Pretrain over more data
- Dynamically change the masking patterns applied to the training data in each epoch

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

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]

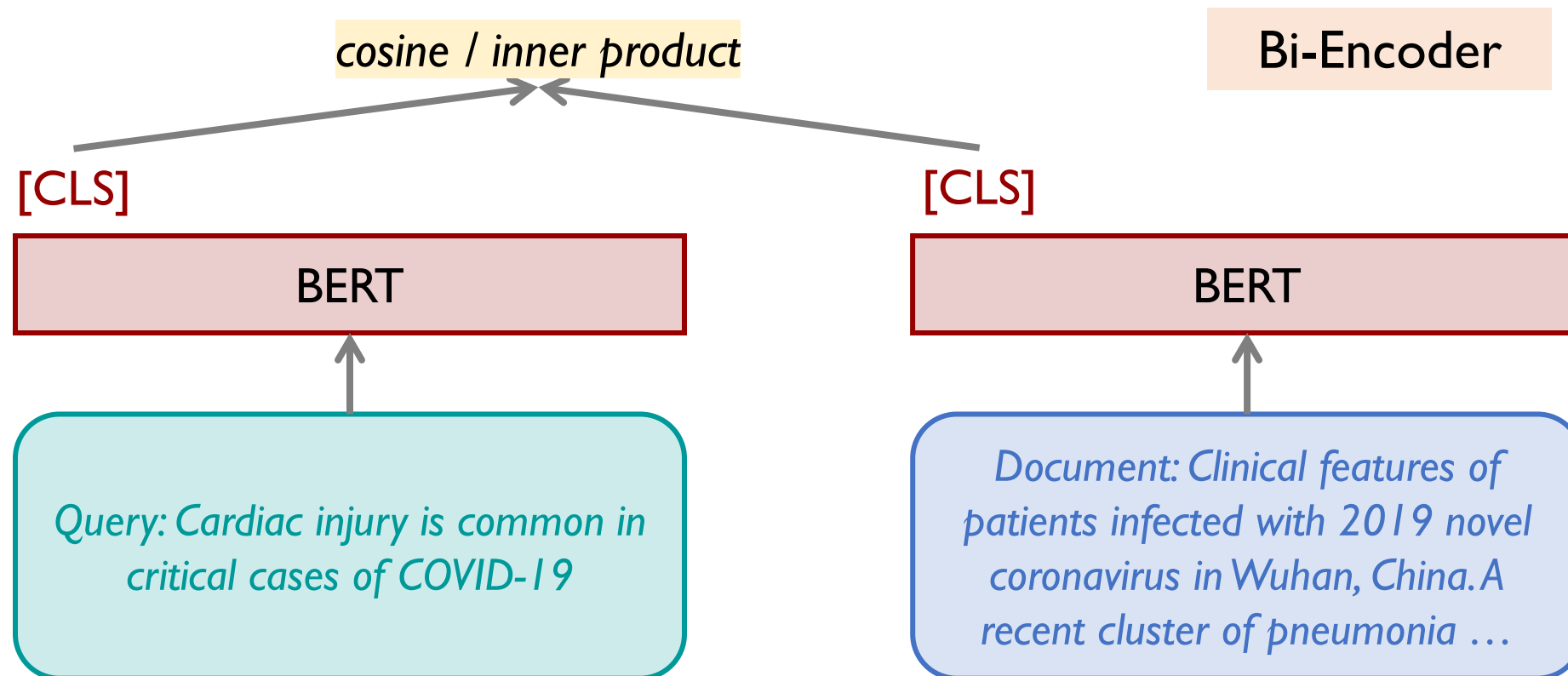
- Use a small MLM as an auxiliary generator (discarded after pretraining)
- Pretrain the main model as a discriminator
- The small auxiliary MLM and the main discriminator are jointly trained.
- The main model's pretraining task becomes **more and more challenging** in pretraining.



BERT-Based Ranking

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

```
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)

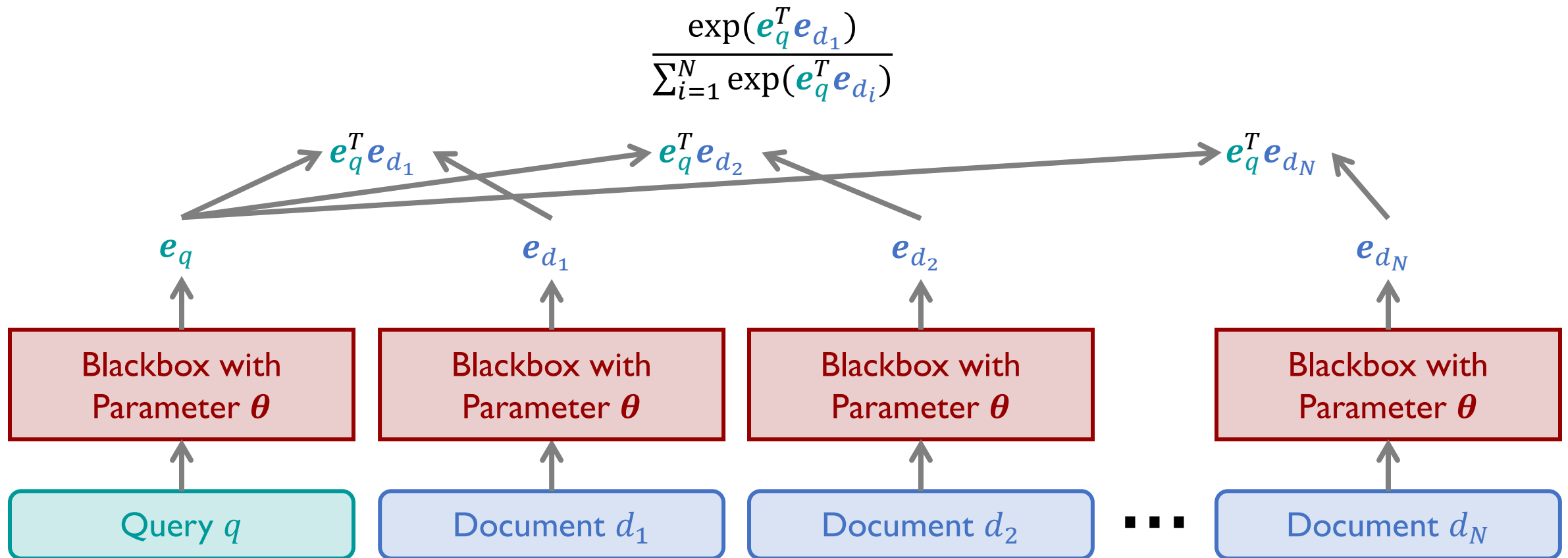
# Extract [CLS] token embedding
cls_embedding = outputs.last_hidden_state[:, 0, :] # shape: [1, 768]
```

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!

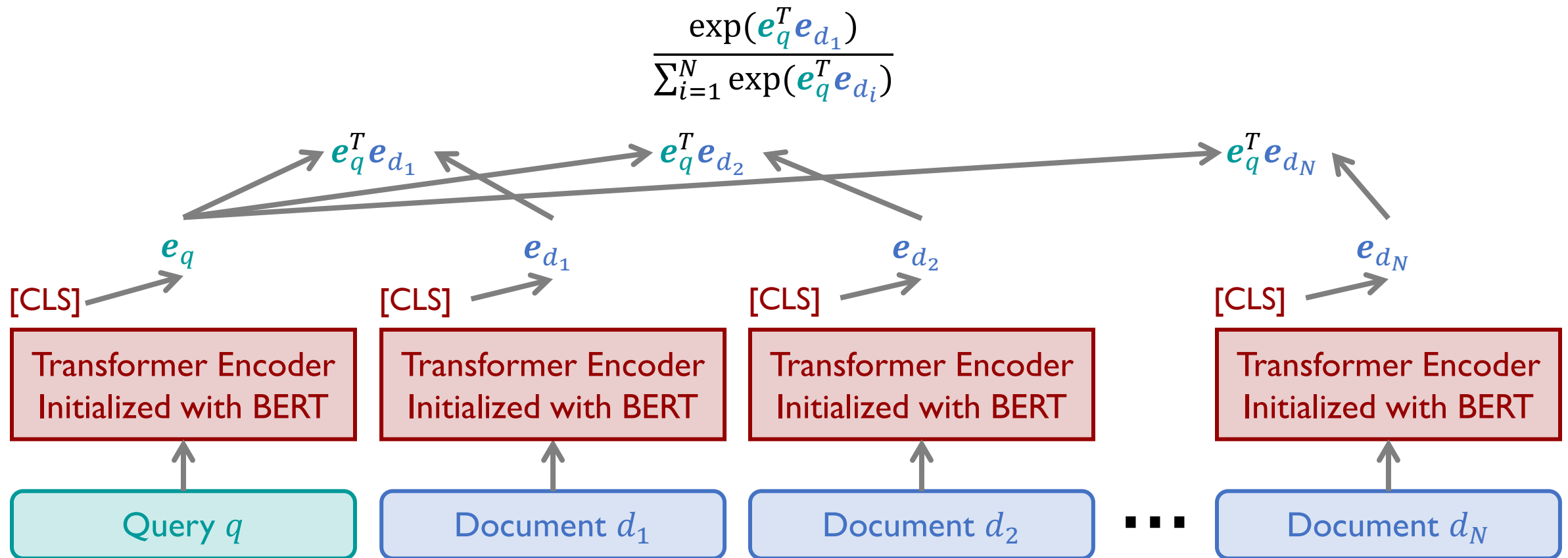
Dense Passage Retrieval [Karpukhin et al., EMNLP 2020]

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



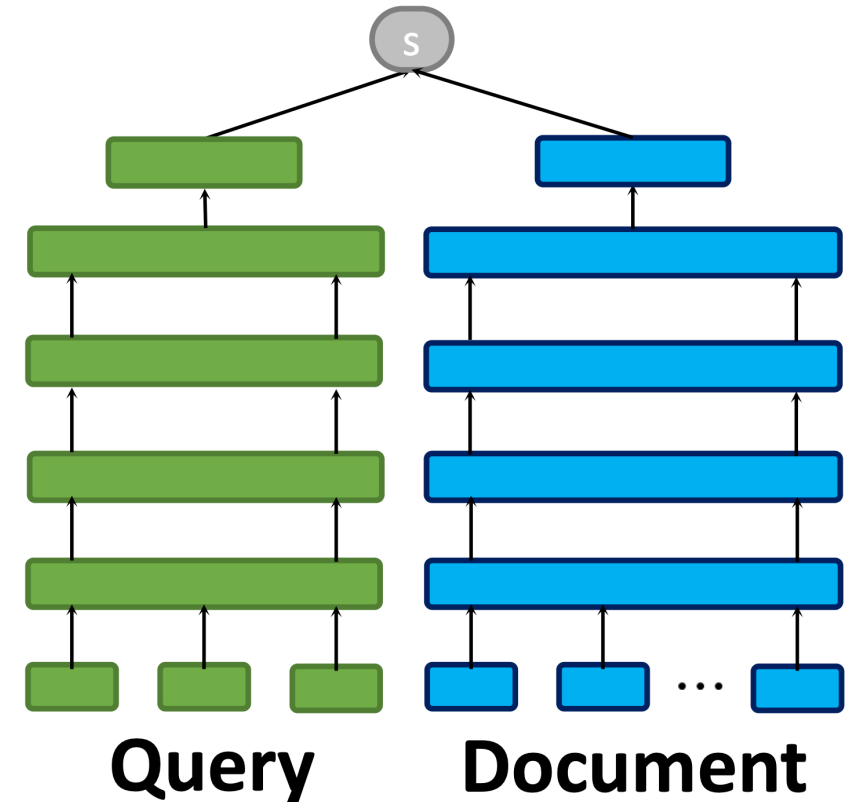
Dense Passage Retrieval (BERT Version)

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



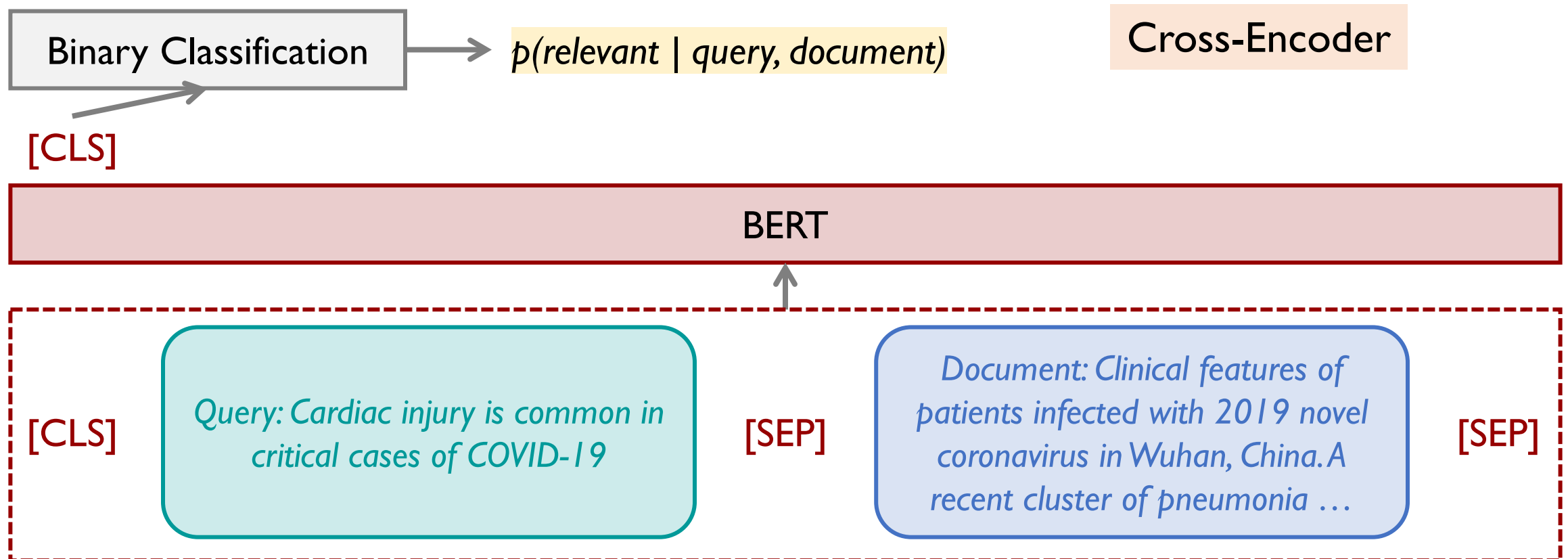
Fine-Tune a Bi-Encoder Ranking Model

- 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**.



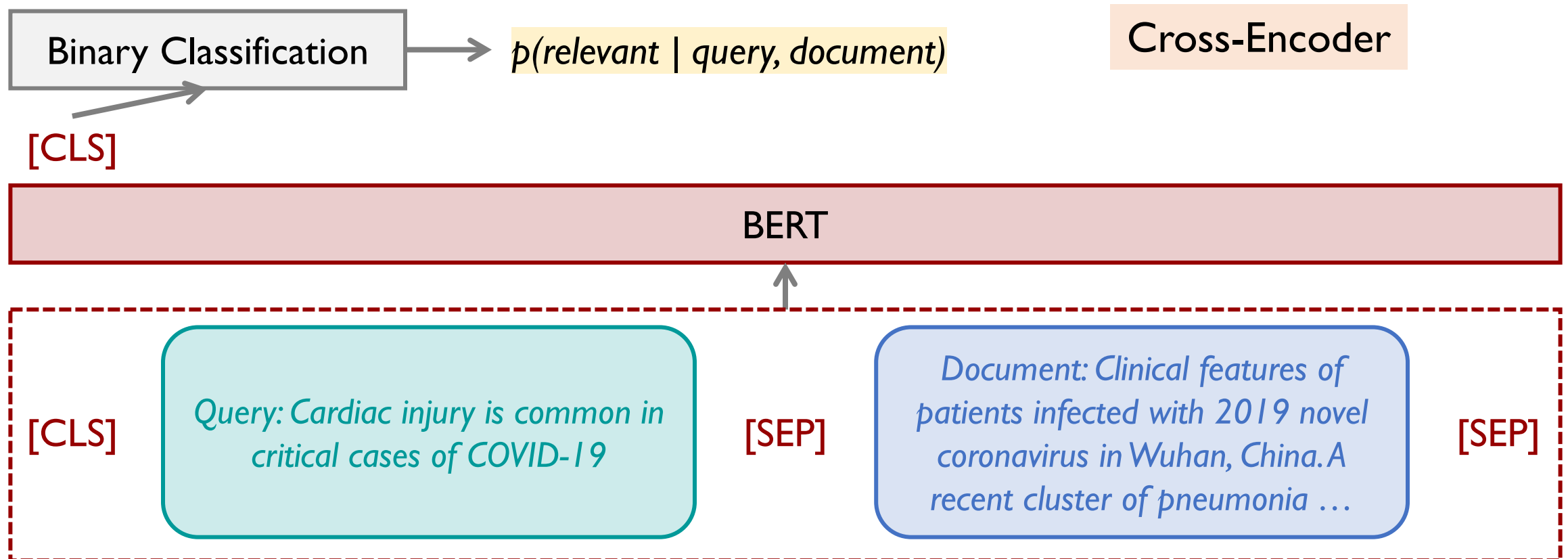
How to use BERT for retrieval? – Solution 2

- 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

$$\text{score}(q, d) = \max_{1 \leq k \leq K} \text{score}(q, p_k)$$

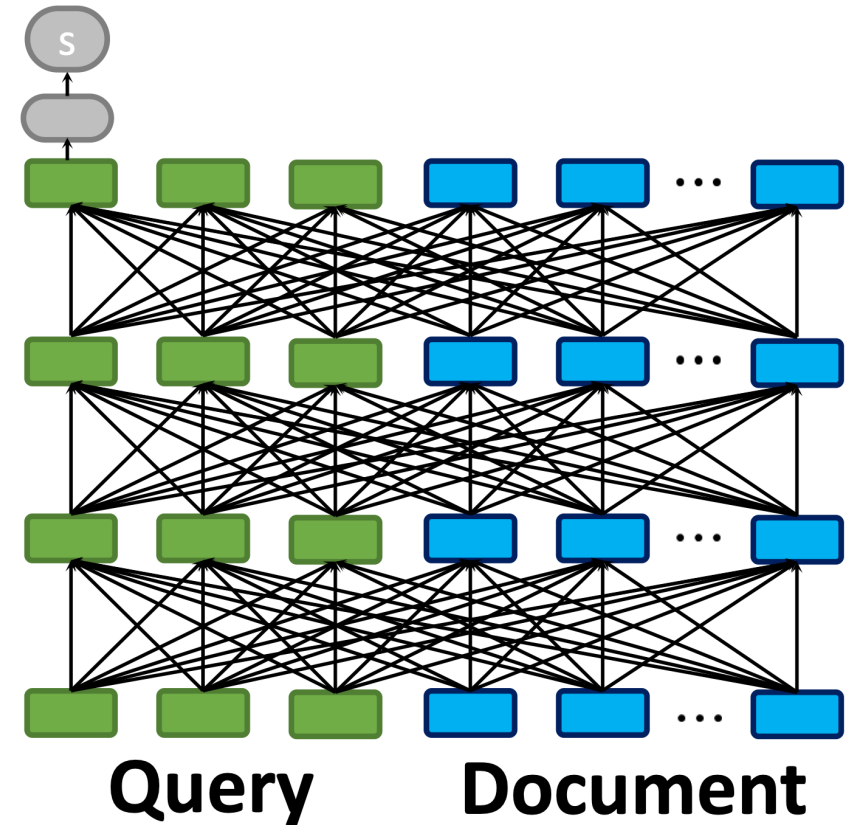
- SumP

$$\text{score}(q, d) = \sum_{k=1}^K \text{score}(q, p_k)$$

Model	nDCG@20			
	Robust04		ClueWeb09-B	
	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 [†]	0.491 [†]	0.286	0.272[†]
BERT-MaxP	0.469[†]	0.529[†]	0.293	0.262 [†]
BERT-SumP	0.467 [†]	0.524 [†]	0.289	0.261

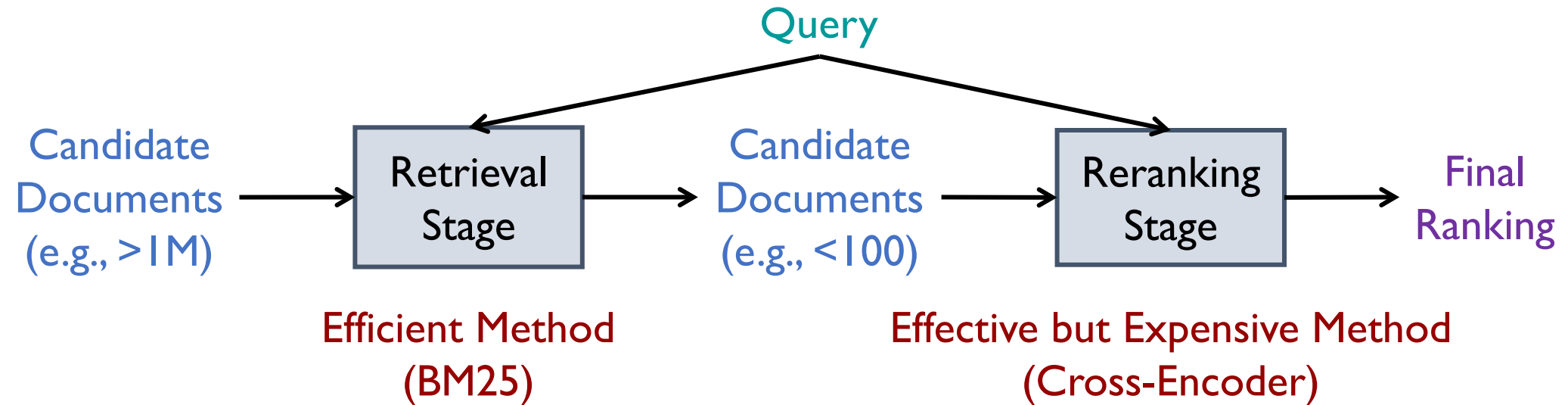
Fine-Tune a Cross-Encoder Ranking Model

- **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		TREC-CAR MAP
	Dev	Eval	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](#)



[X 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)

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

- 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]
 - $\text{MRR} = \frac{1}{3} \times \left(\frac{1}{4} + \frac{1}{1} + \frac{1}{2} \right) = \frac{7}{12} \approx 0.583$



Thank You!

Course Website: <https://yuzhang-teaching.github.io/CSCE670-F26.html>