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

Source : NAACL-HLT 2019

Speaker : Ya-Fang, Hsiao

Advisor : Jia-Ling, Koh

Date : 2019/09/02

CONTENTS

Introduction

Introduction

Bidirectional **E**ncoder **R**epresentations from **T**ransformers

Language Model

$$P(w_1, w_2, \dots, w_T) = \prod_{t=1}^{T} P(w_t | w_1, w_2, \dots, w_{t-1})$$

Pre-trained Language Model

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

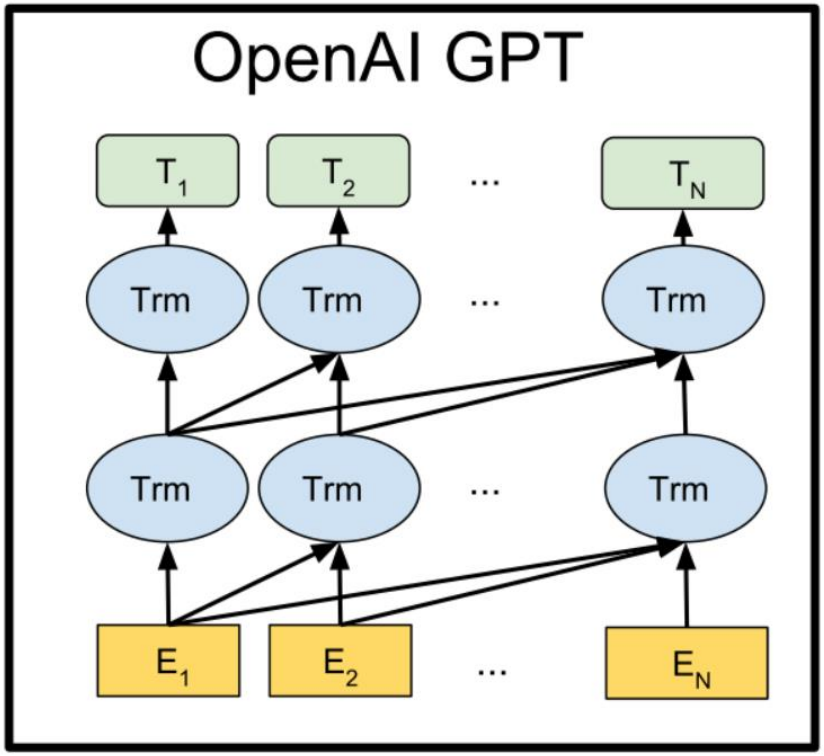
2 Related Work

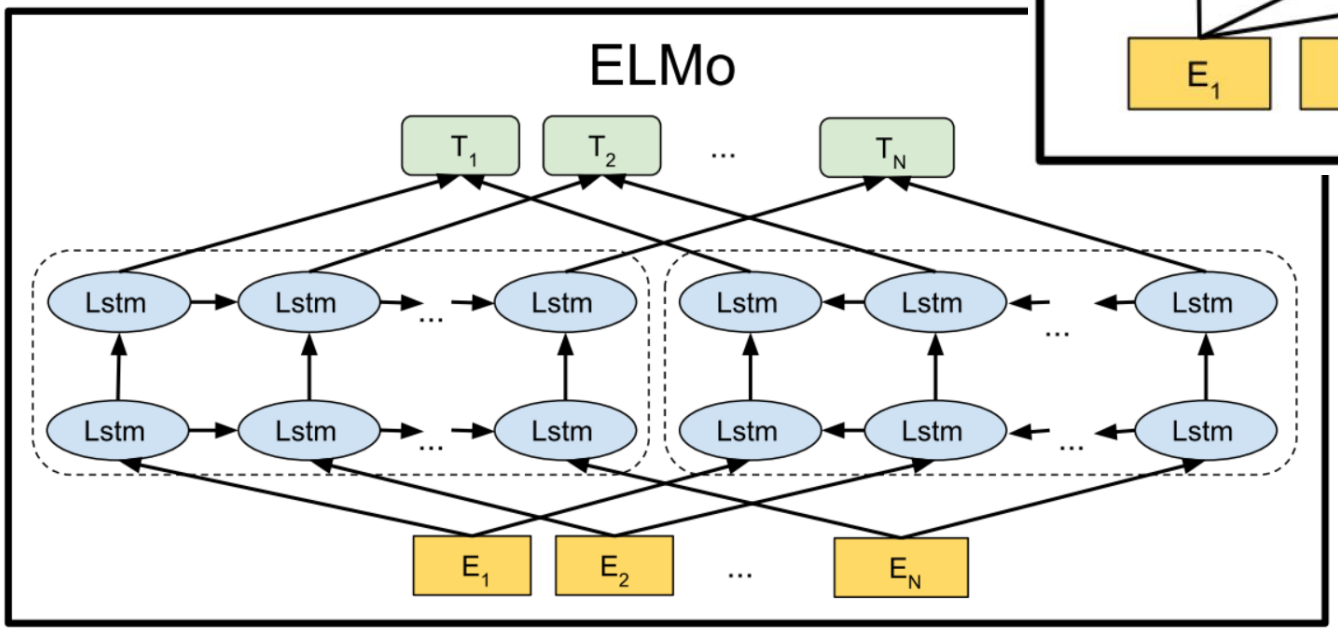
Related Work

Pre-trained Language Model

Feature-based : ELMo

Fine-tuning : OpenAI GPT





Related Work

Pre-trained Language Model

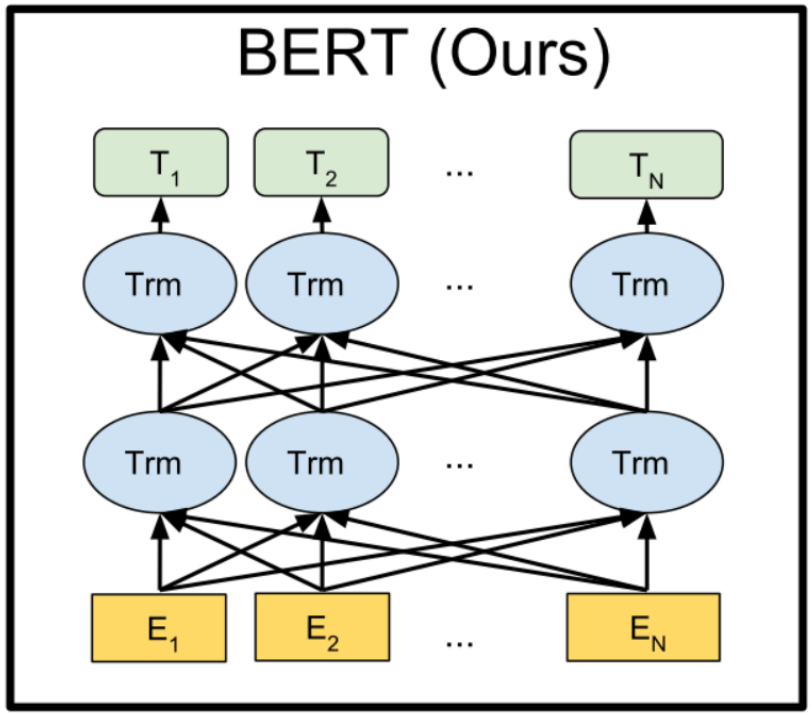
Feature-based : ELMo 

Fine-tuning : OpenAI GPT

1. Unidirectional language model
2. Same objective function

Bidirectional **E**ncoder **R**epresentations from **T**ransformers

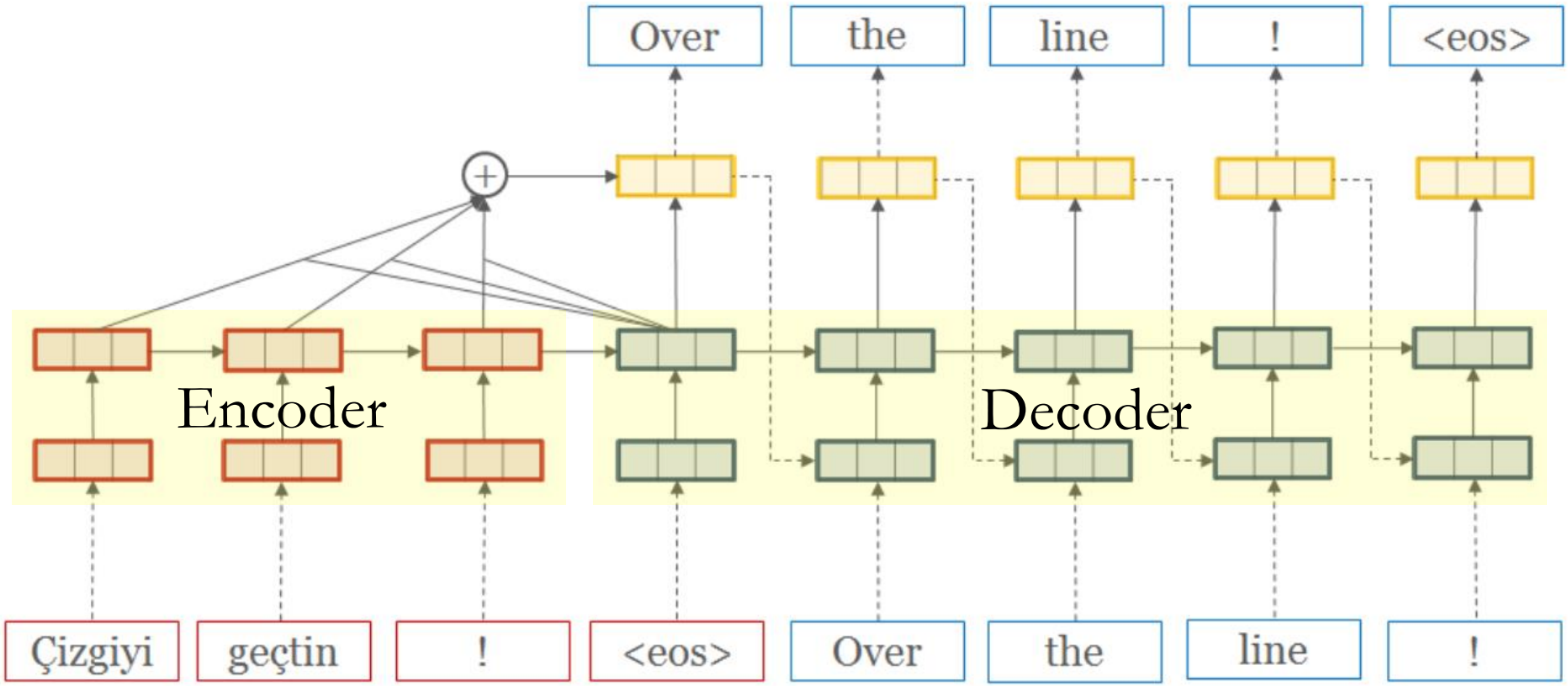
Masked Language Models (MLM)
Next Sentence Prediction (NSP)



Transformers

《Attention is all you need》
Vaswani et al. (NIPS2017)

Sequence2sequence

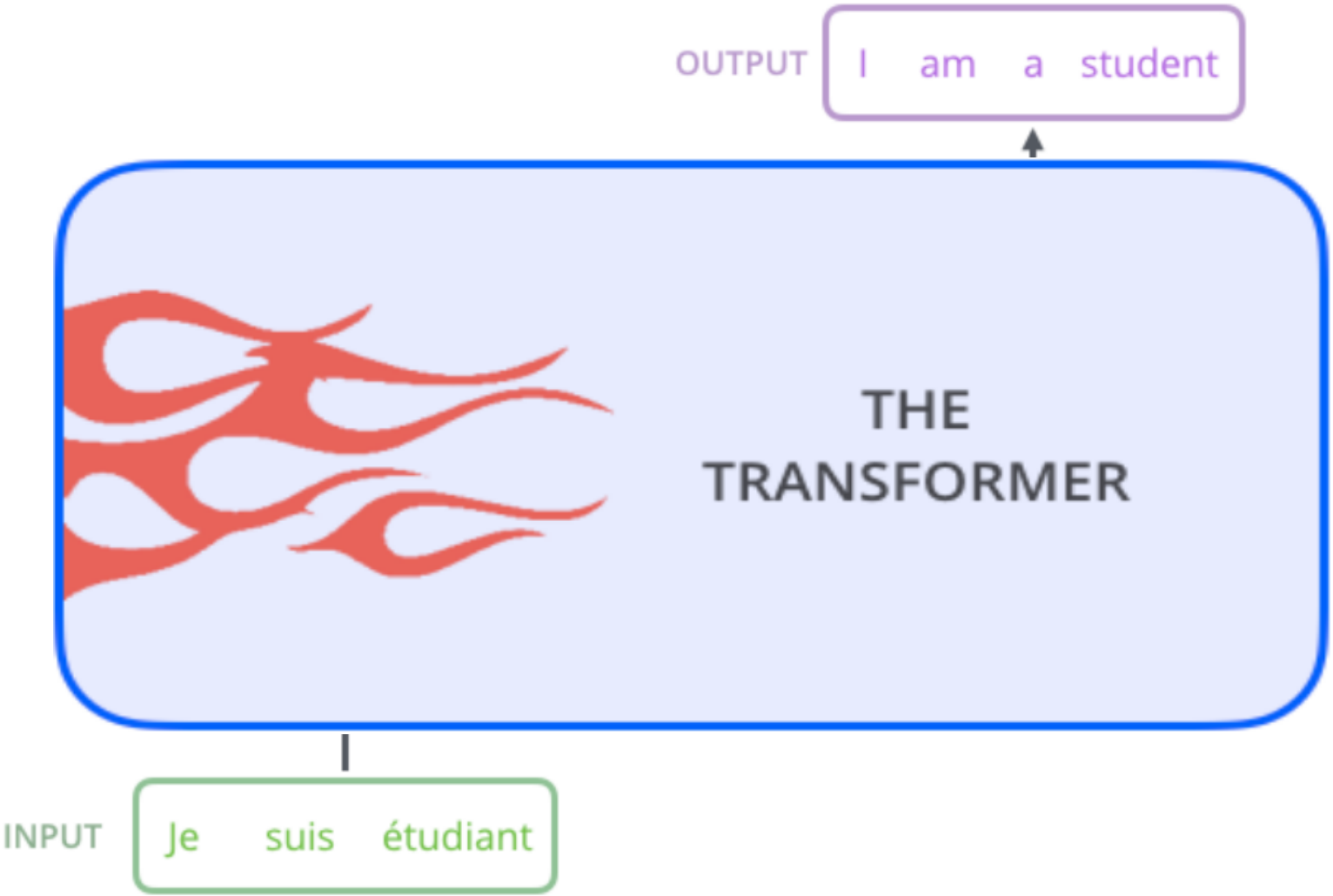


RNN : hard to parallel

Transformers

《Attention is all you need》
Vaswani et al. (NIPS2017)

Encoder-Decoder

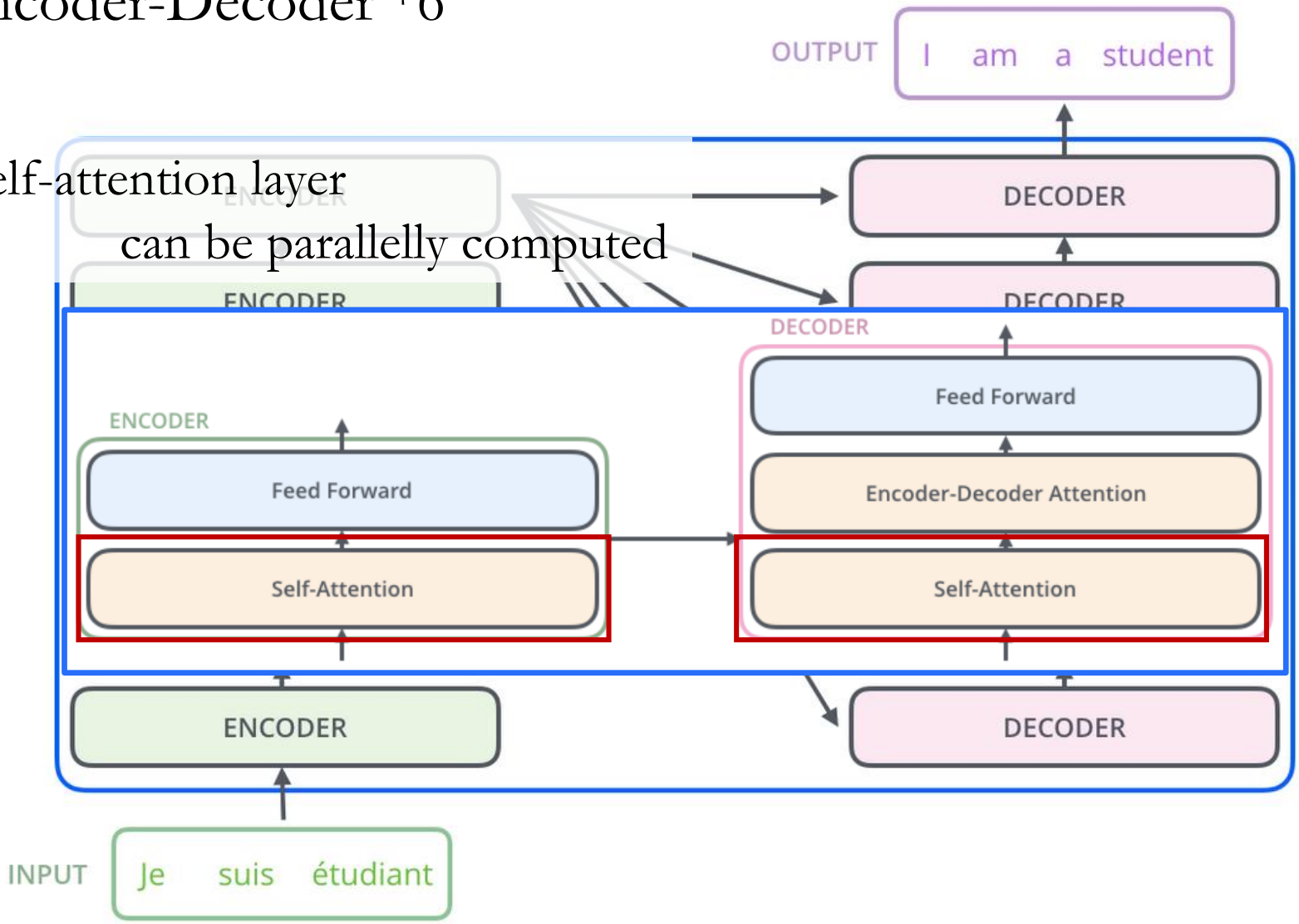


Transformers

《Attention is all you need》
Vaswani et al. (NIPS2017)

Encoder-Decoder *6

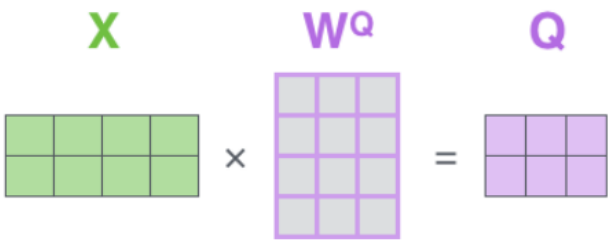
Self-attention layer
can be parallelly computed



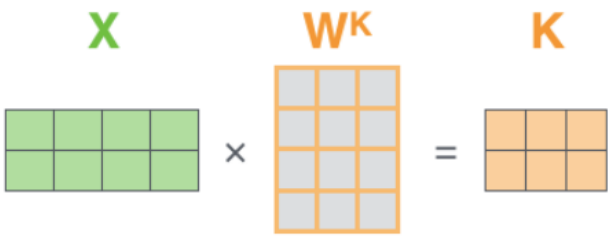
Transformers

《Attention is all you need》
Vaswani et al. (NIPS2017)

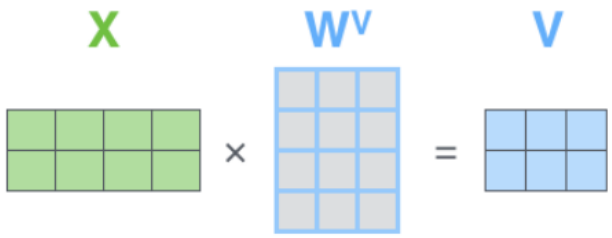
Self-Attention



query (to match others)

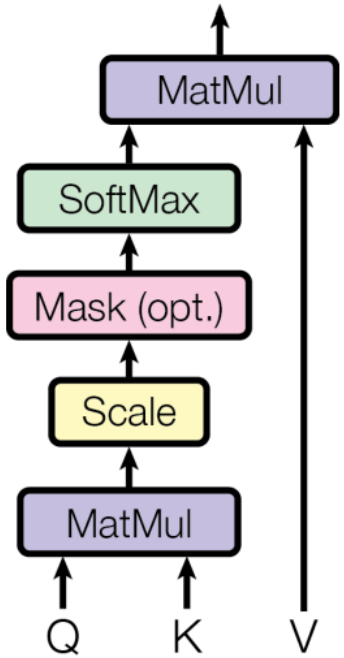


key (to be matched)



information to be extracted

Scaled Dot-Product Attention

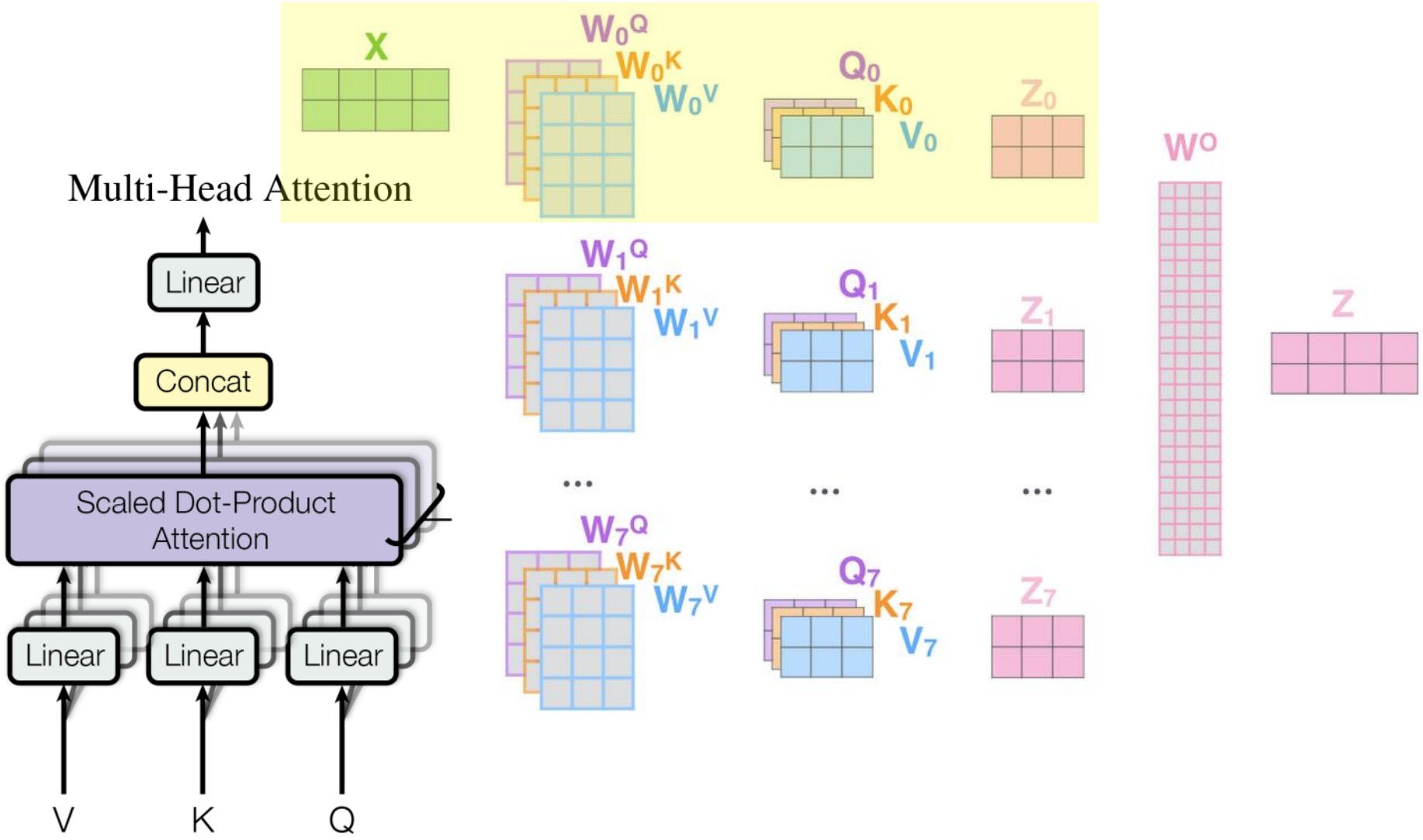


$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$

Transformers

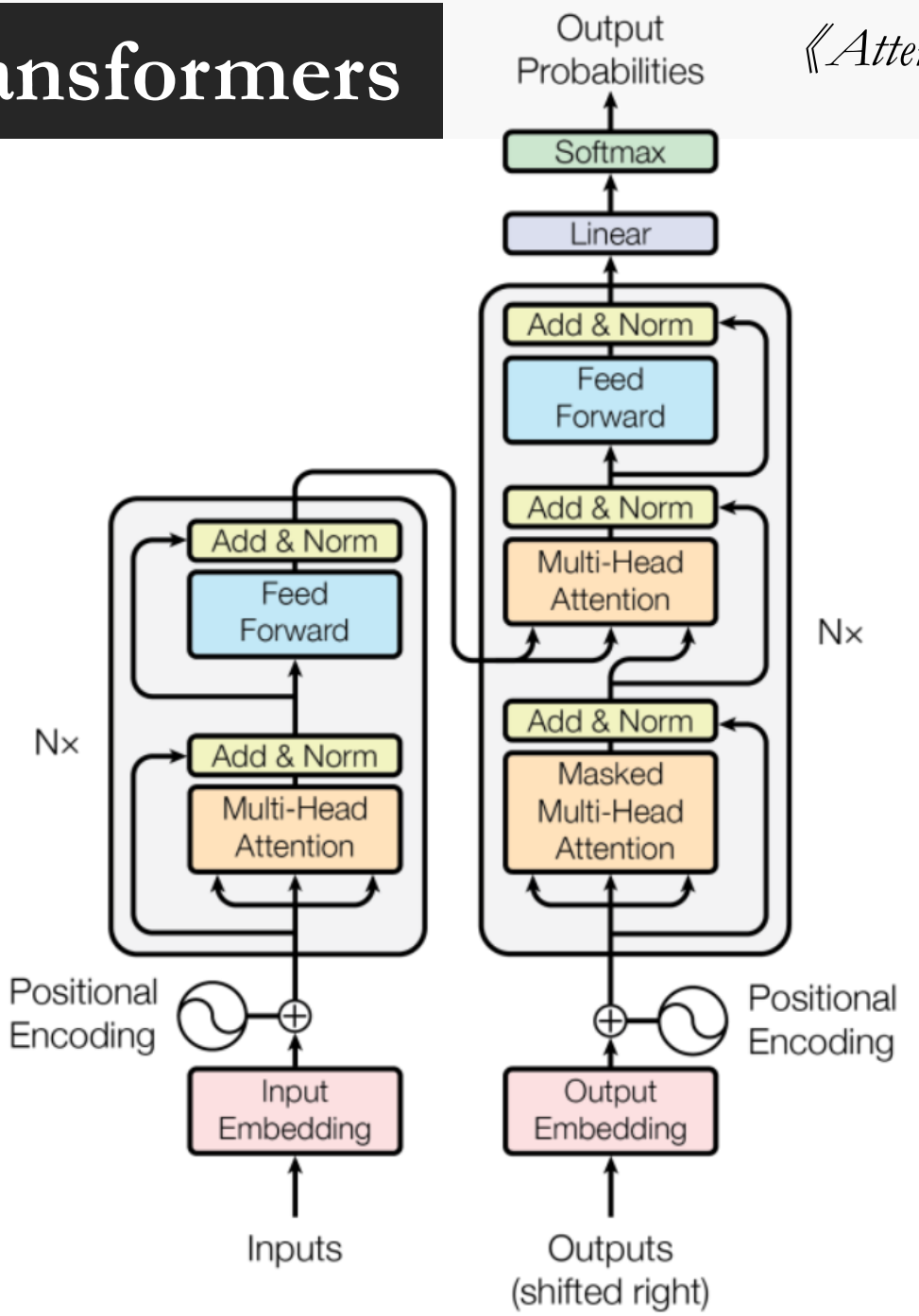
《Attention is all you need》
Vaswani et al. (NIPS2017)

Multi-Head Attention



Transformers

《Attention is all you need》
Vaswani et al. (NIPS2017)

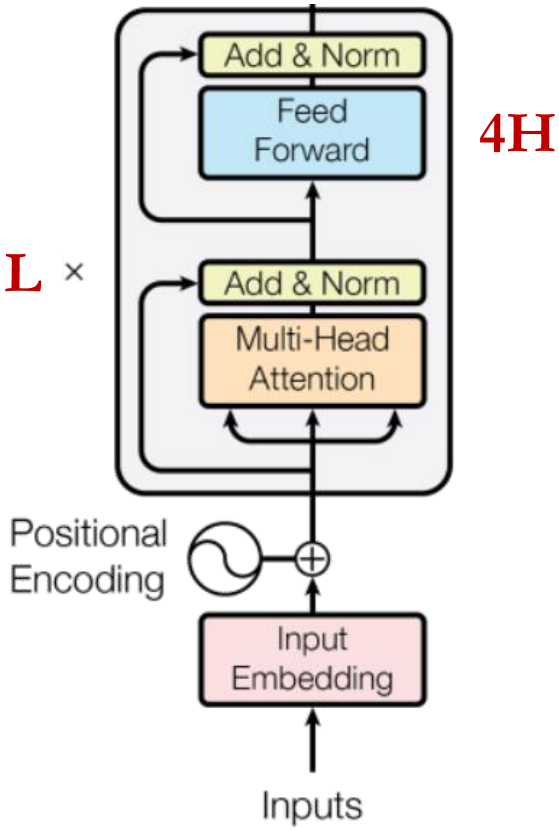


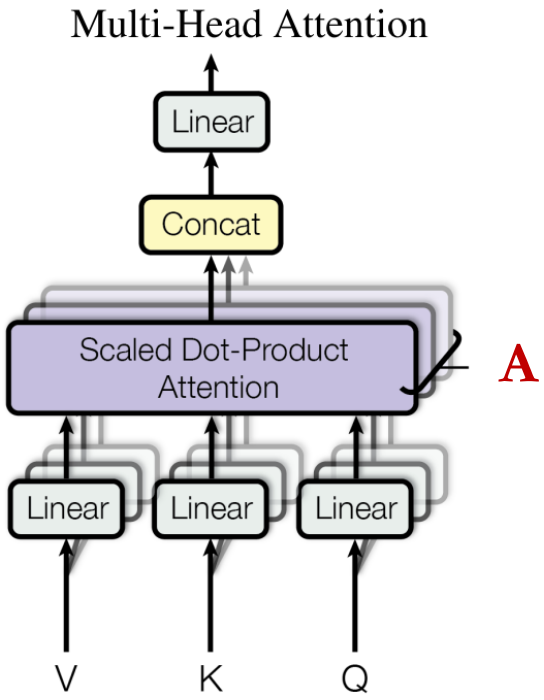
BERT

Bidirectional **E**ncoder **R**epresentations from **T**ransformers

$BERT_{BASE}$
(L=12, H=768, A=12, Parameters=110M)

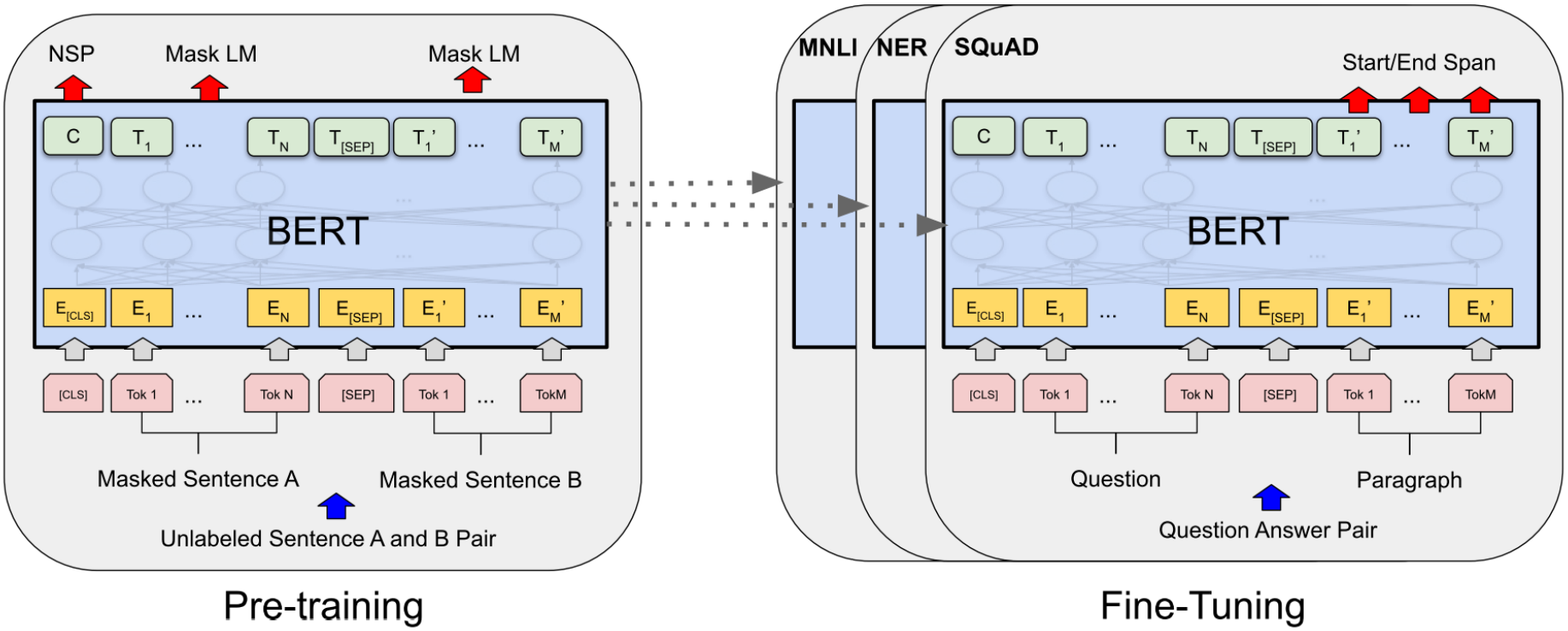
$BERT_{LARGE}$
(L=24, H=1024, A=16, Parameters=340M)





Method 3

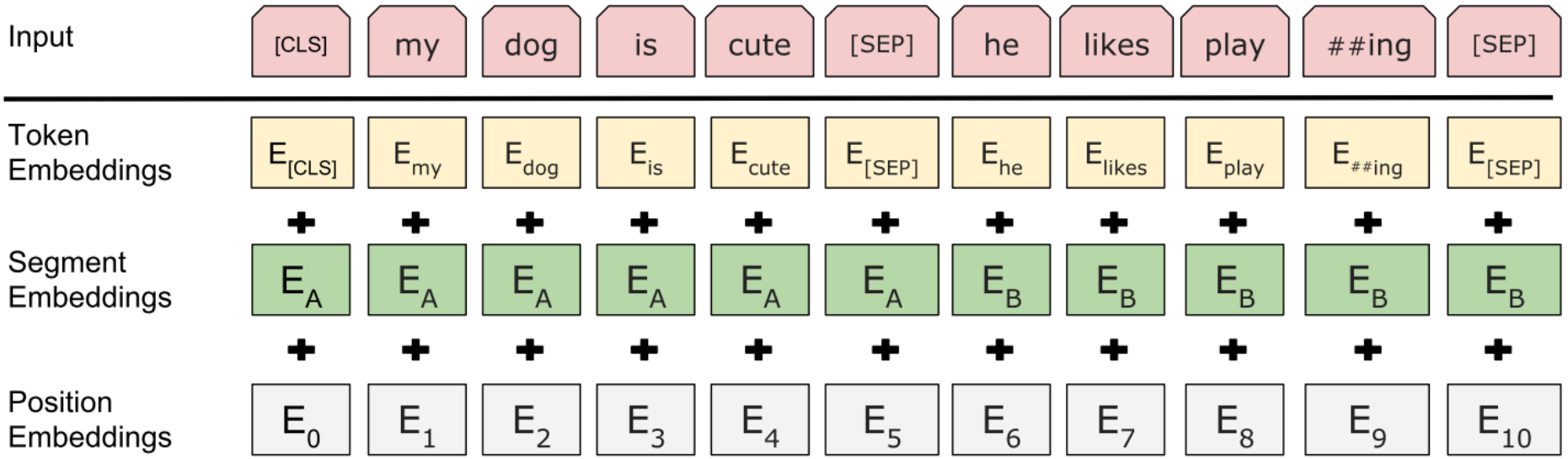
Framework



Pre-training : trained on unlabeled data over different pre-training tasks.

Fine-Tuning : fine-tuned parameters using labeled data from the downstream tasks.

Input



[CLS] : classification token

[SEP] : separate token

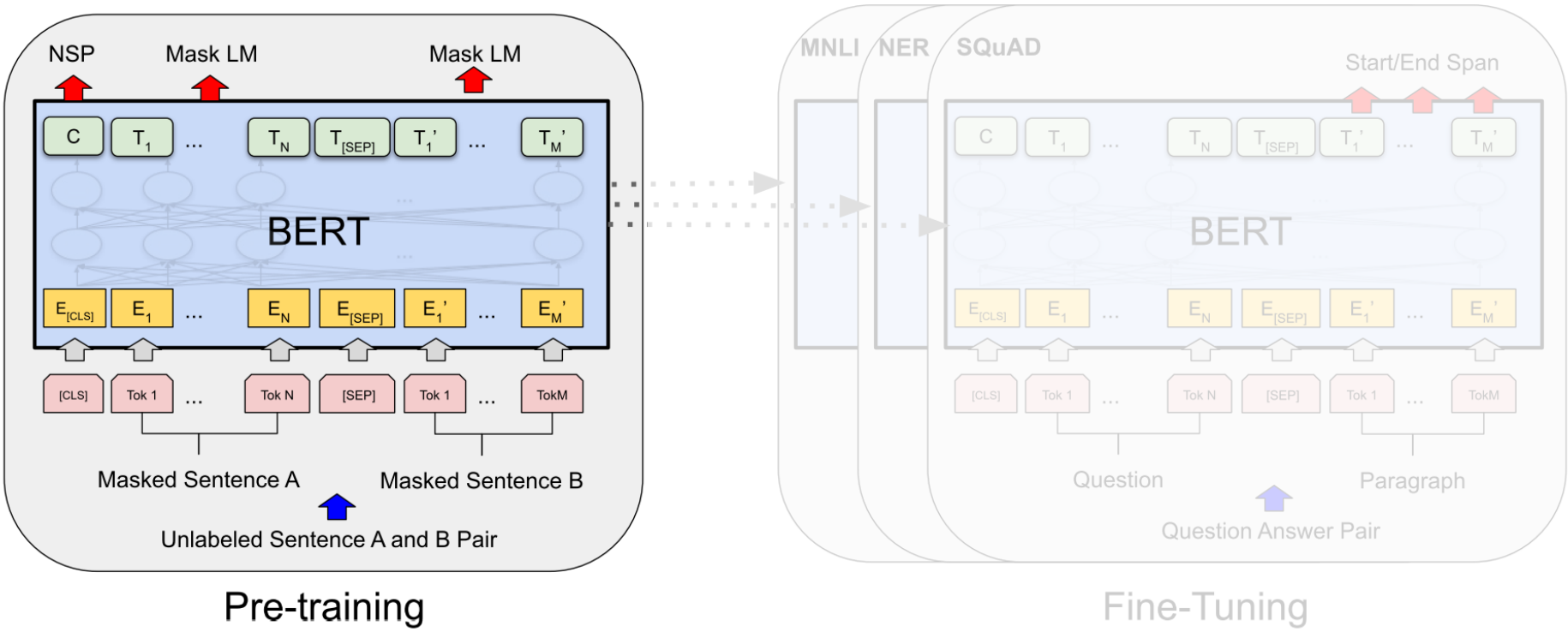
Pre-training corpus : BooksCorpus、English Wikipedia

Token Embedding : WordPiece embeddings with a 30,000 token vocabulary.

Segment Embedding : Learned embeddings belong to sentence A or sentence B.

Position Embedding : Learned positional embeddings.

Pre-training

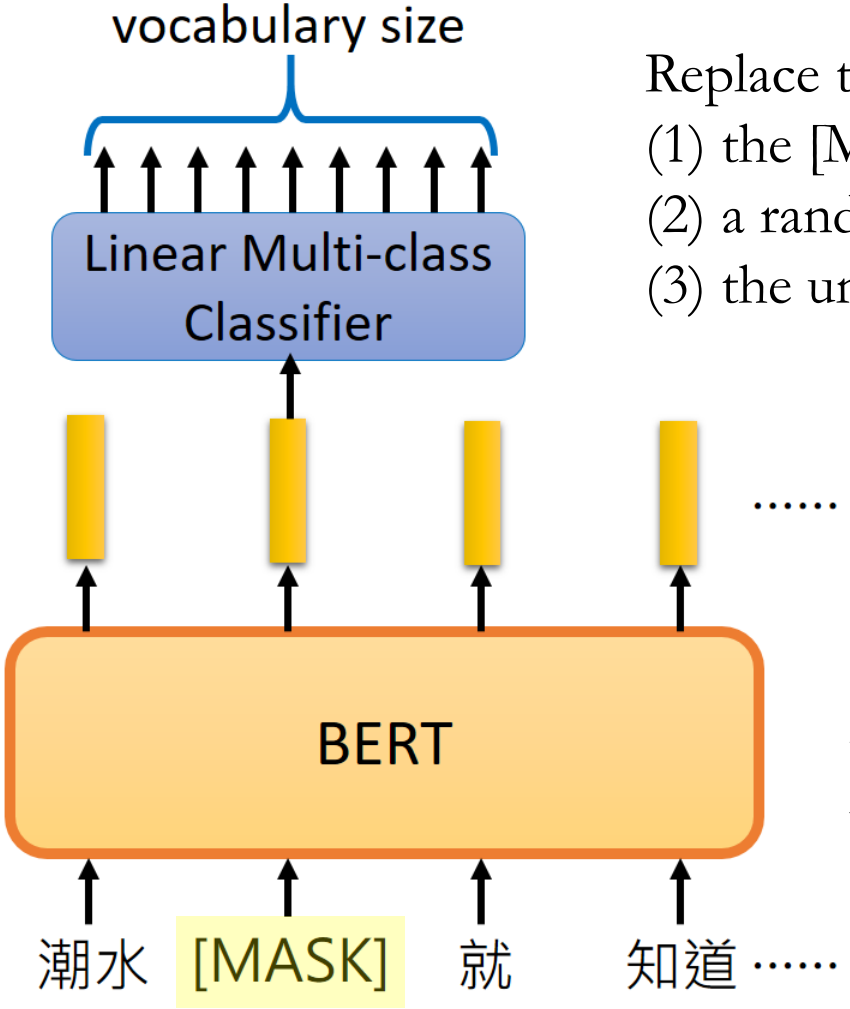


Two unsupervised tasks:

1. Masked Language Models (MLM)
2. Next Sentence Prediction (NSP)

Task1. MLM

Masked Language Models



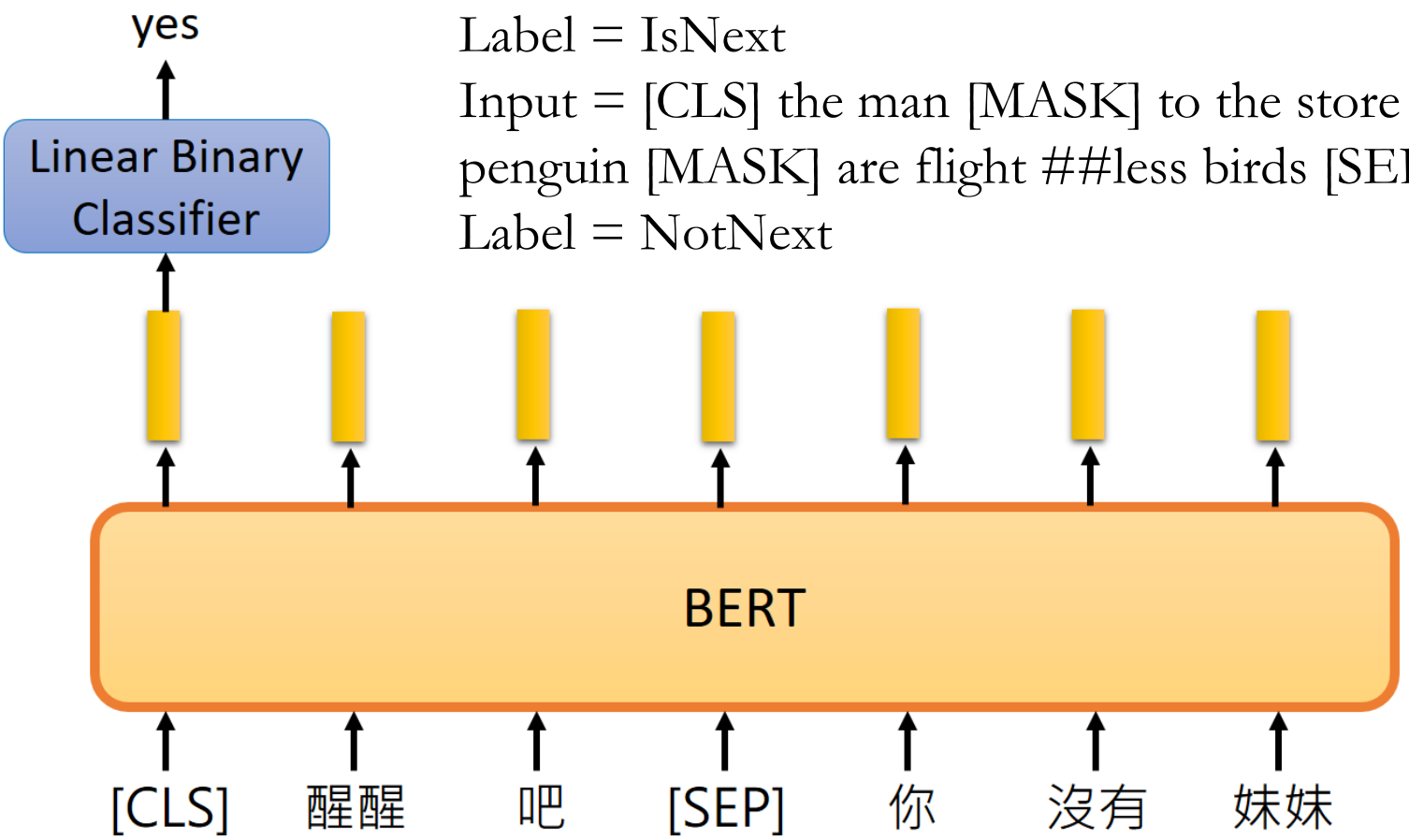
Replace the token with
(1) the [MASK] token 80% of the time.
(2) a random token 10% of the time.
(3) the unchanged i-th token 10% of the time.

Mask 15% of all WordPiece tokens in each sequence at random for prediction.

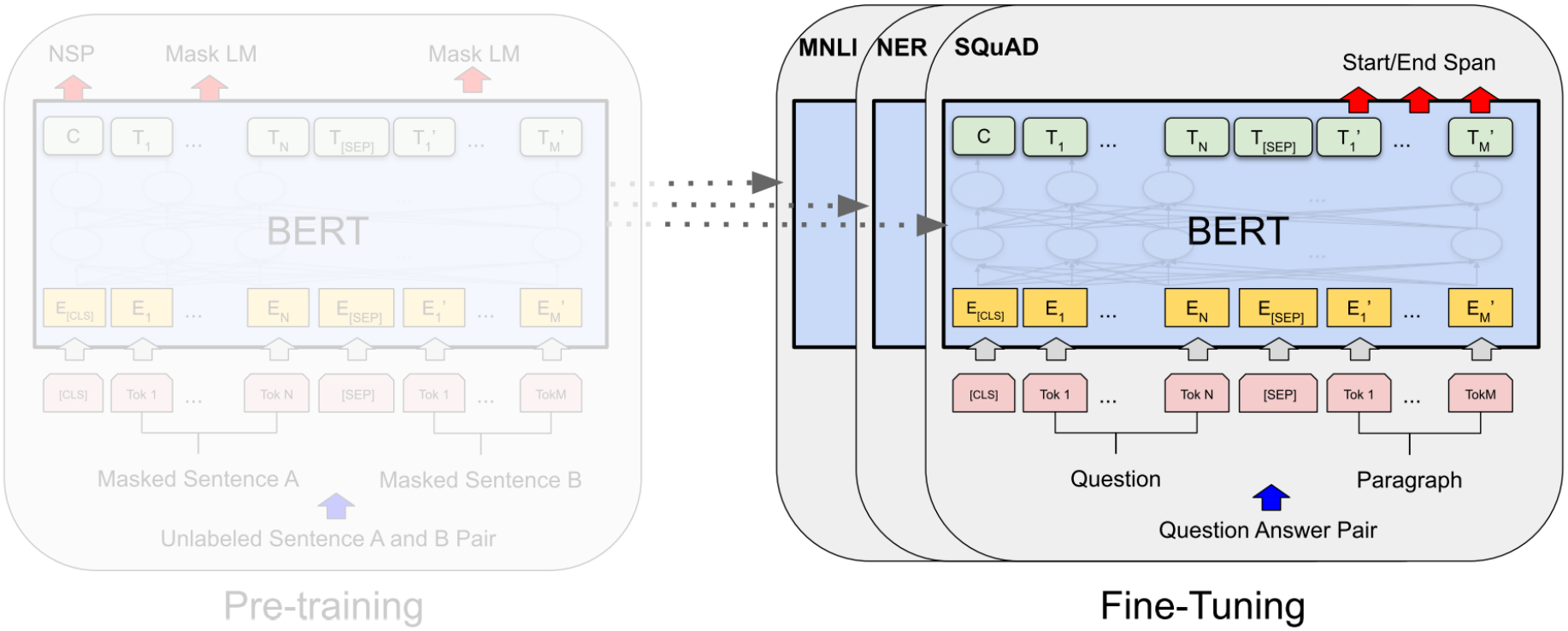
Task2. NSP

Next Sentence Prediction

Input = [CLS] the man went to [MASK] store [SEP]
he bought a gallon [MASK] milk [SEP]
Label = IsNext
Input = [CLS] the man [MASK] to the store [SEP]
penguin [MASK] are flight ##less birds [SEP]
Label = NotNext



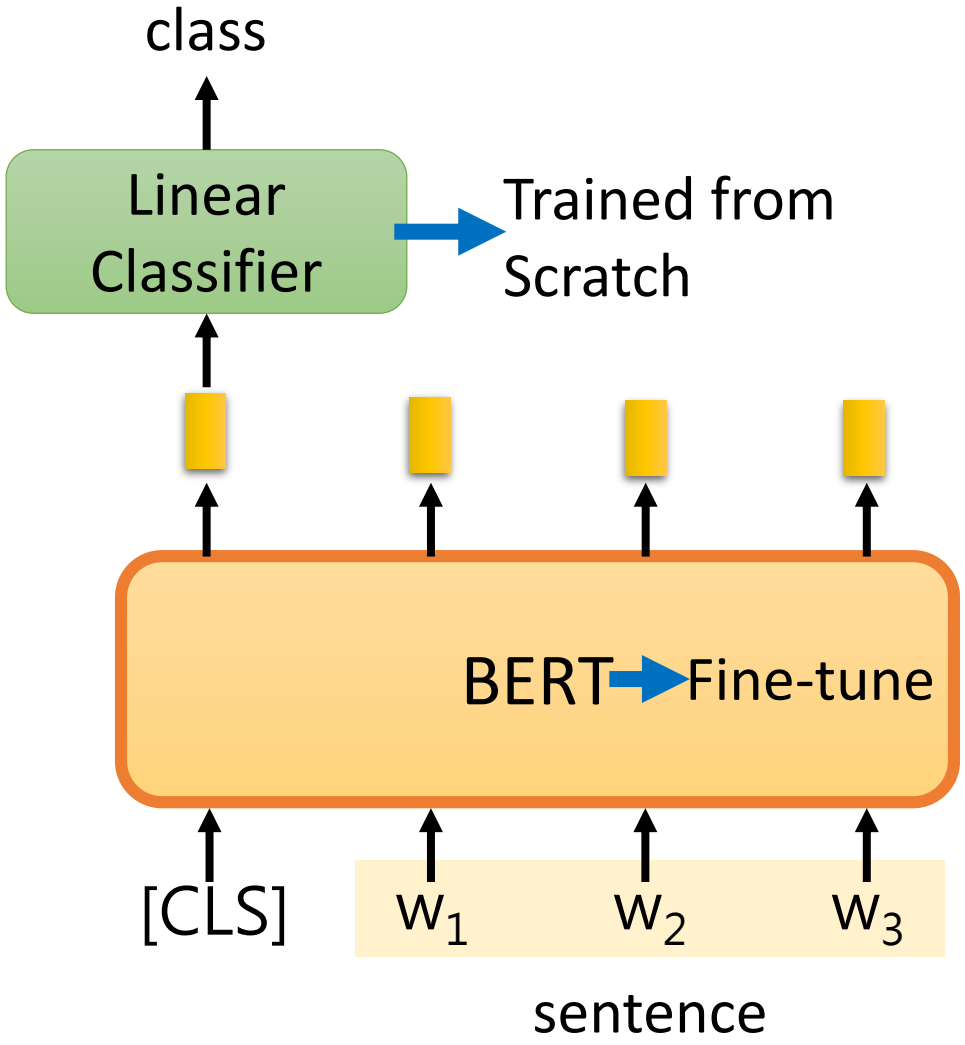
Fine-Tuning



Fine-Tuning : fine-tuned parameters using labeled data from the downstream tasks.

Task 1 (b)

Single Sentence Classification Tasks

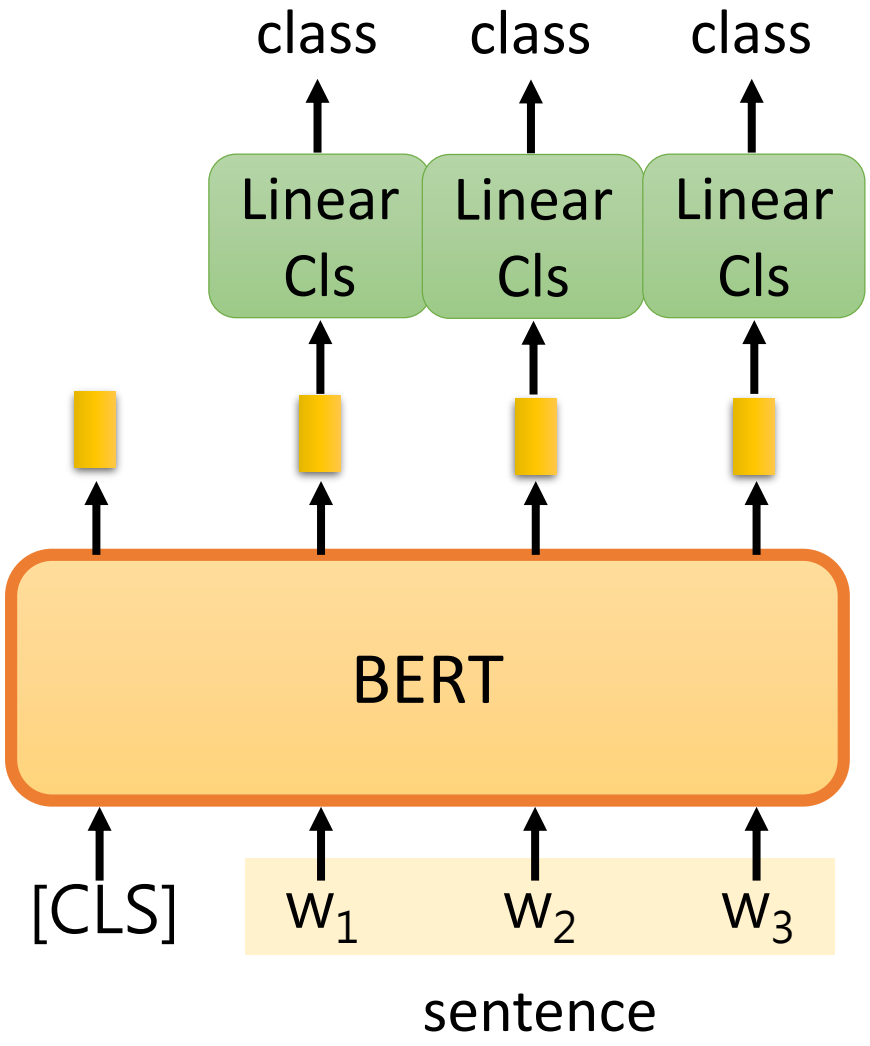


Input: single sentence, output: class

Example:
Sentiment analysis
Document Classification

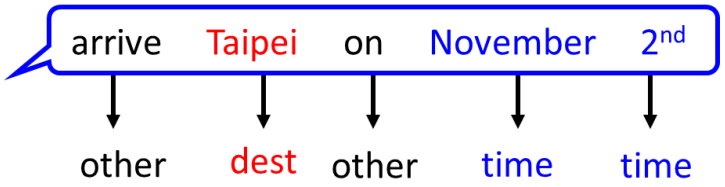
Task 2 (d)

Single Sentence Tagging Tasks



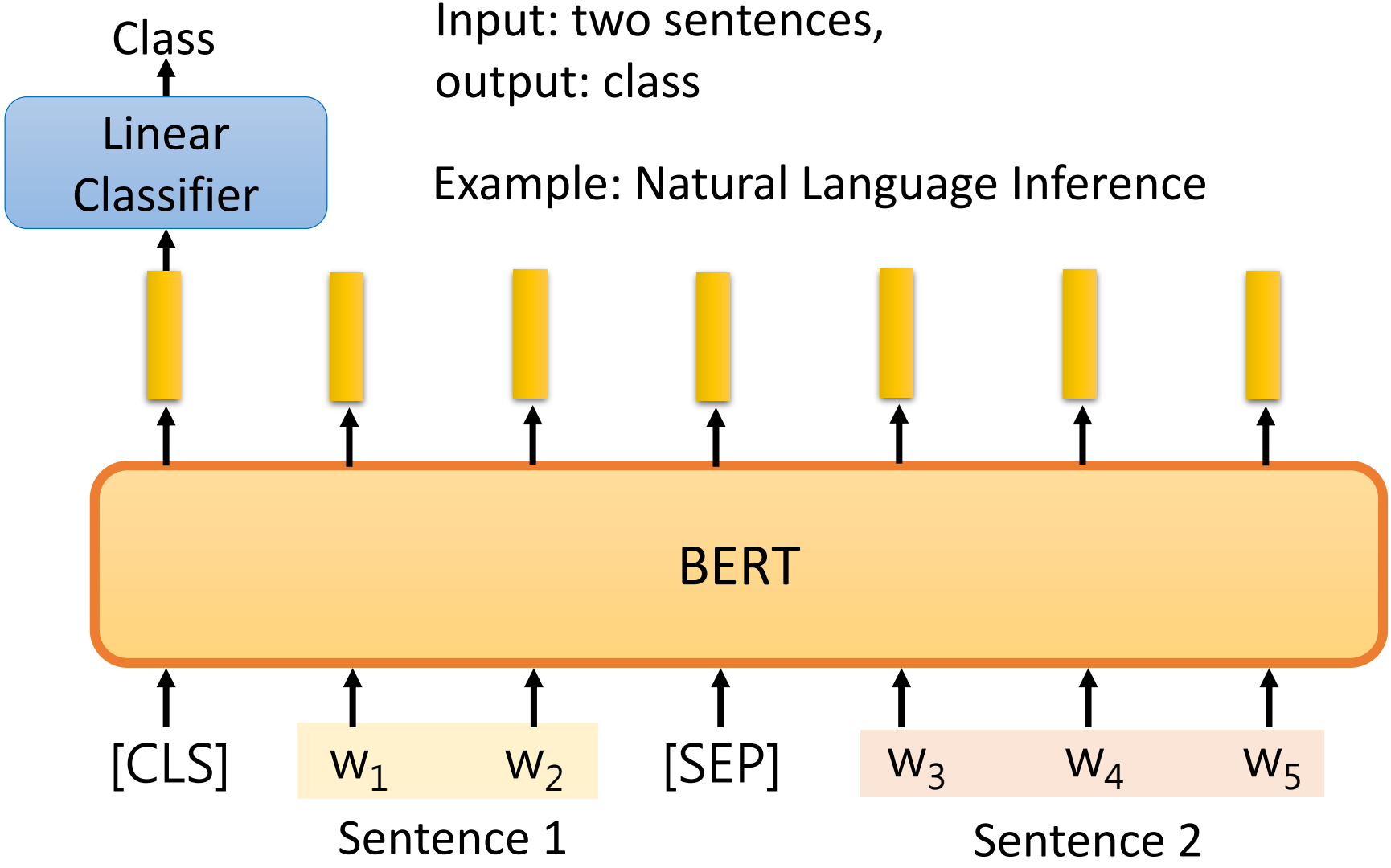
Input: single sentence, output: class of each word

Example: Slot filling



Task 3 (a)

Sentence Pair Classification Tasks

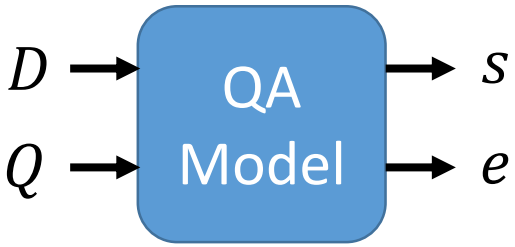


Task 4 (c)

Question Answering Tasks

Document: $D = \{d_1, d_2, \cdots, d_N\}$

Query: $Q = \{q_1, q_2, \cdots, q_N\}$



output: two integers (s, e)

Answer: $A = \{q_s, \cdots, q_e\}$

In meteorology, precipitation is any product of the condensation of 17 spheric water vapor that falls under gravity. The main forms of precipitation include drizzle, rain, sleet, snow, graupel and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. Short, intense periods of rain 77 atte 79 cations are called "showers".

What causes precipitation to fall?
gravity

$s = 17, e = 17$

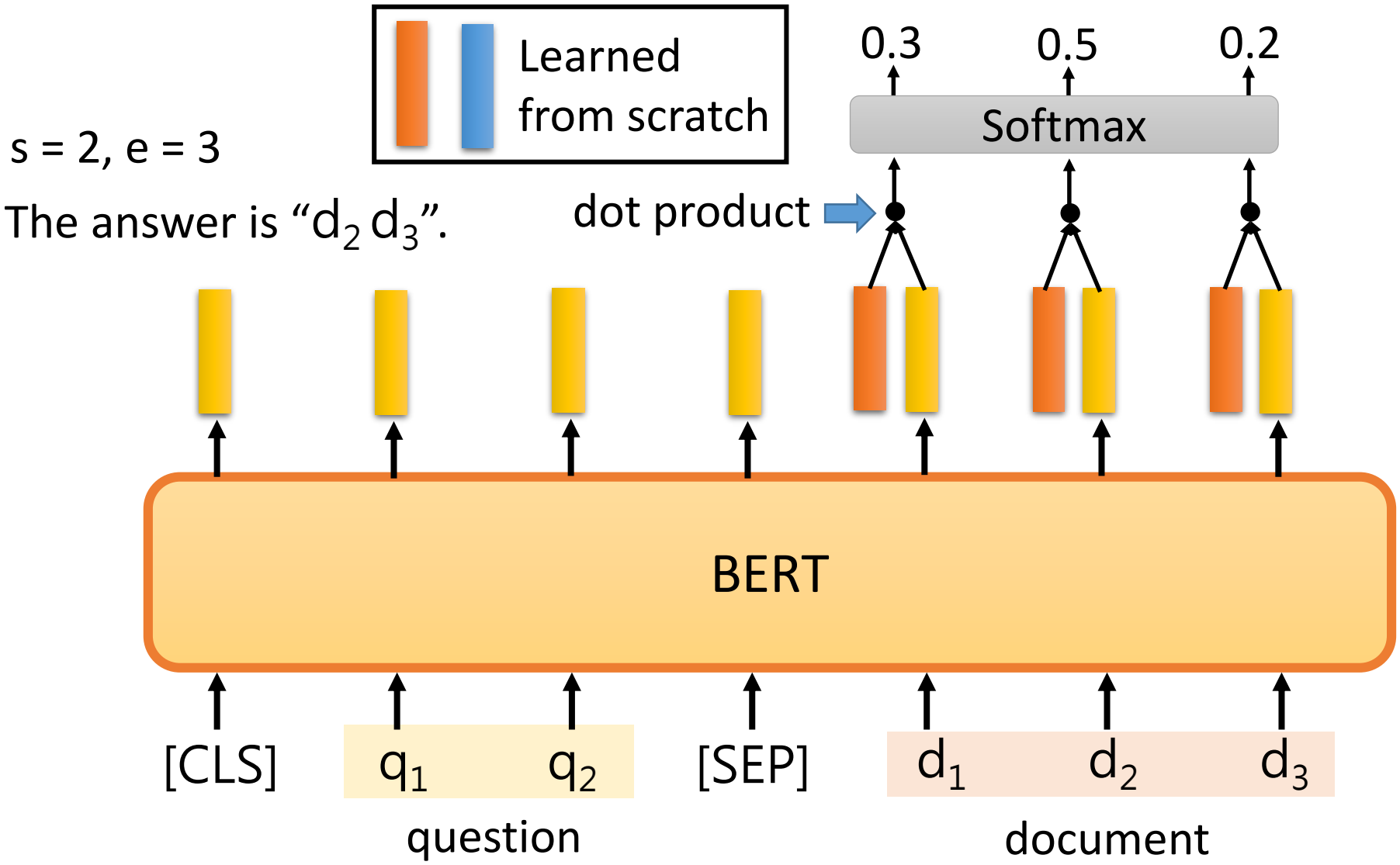
What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

$s = 77, e = 79$

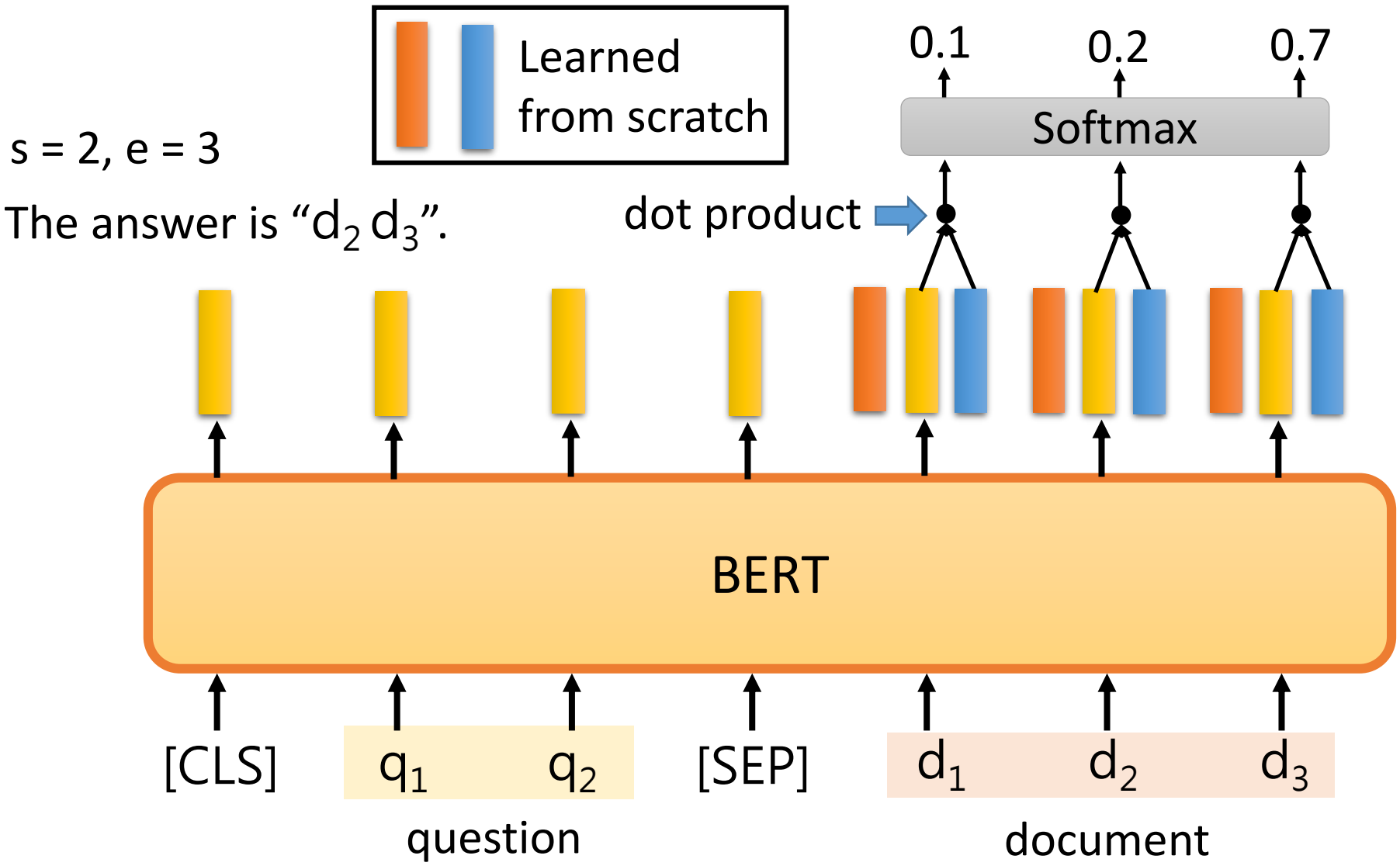
Task 4 (c)

Question Answering Tasks



Task 4 (c)

Question Answering Tasks



Experiment

Experiments

Fine-tuning results on 11 NLP tasks

Implements

LeeMeng-進擊的BERT (Pytorch)

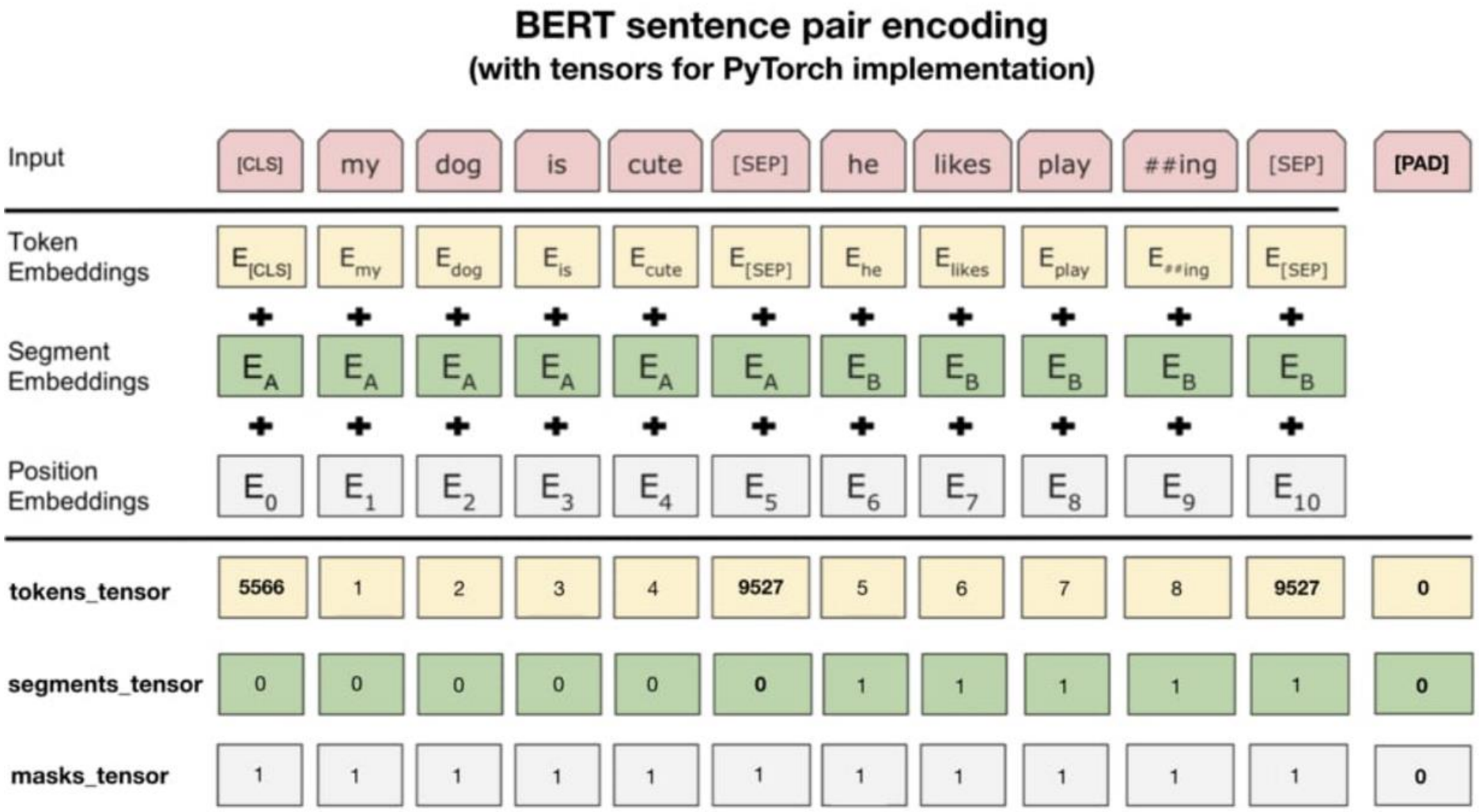
```
GITHUB_REPO = "huggingface/pytorch-pretrained-BERT" # 感謝 HuggingFace 團隊造福後人
PRETRAINED_MODEL_NAME = "bert-base-chinese"  # 指定繁簡中文 BERT-BASE 預訓練模型

# 取得此預訓練模型所使用的 tokenizer
tokenizer = torch.hub.load(GITHUB_REPO, 'bertTokenizer', PRETRAINED_MODEL_NAME)
```

```
text = "[CLS] 等到潮水 [MASK] 了，就知道誰沒穿褲子。"
tokens = tokenizer.tokenize(text)
ids = tokenizer.convert_tokens_to_ids(tokens)
[CLS] 等到潮水 [MASK] 了，就知道誰沒穿褲子。
['[CLS]', '等', '到', '潮', '水', '[MASK]', '了', '，', '就', '知'] ...
[101, 5023, 1168, 4060, 3717, 103, 749, 8024, 2218, 4761] ...
```

- `[CLS]`：在做分類任務時其最後一層的 repr. 會被視為整個輸入序列的 repr.
- `[SEP]`：有兩個句子的文本會被串接成一個輸入序列，並在兩句之間插入這個 token 以做區隔
- `[UNK]`：沒出現在 BERT 字典裡頭的字會被這個 token 取代
- `[PAD]`：zero padding 遮罩，將長度不一的輸入序列補齊方便做 batch 運算
- `[MASK]`：未知遮罩，僅在預訓練階段會用到

Implements

LeeMeng-進擊的BERT (Pytorch)



Implements

LeeMeng-進擊的BERT (Pytorch)

```
"""
實作可以一次回傳一個 mini-batch 的 DataLoader
這個 DataLoader 吃我們上面定義的 `FakeNewsDataset`，
回傳訓練 BERT 時會需要的 4 個 tensors：
- tokens_tensors  : (batch_size, max_seq_len_in_batch)
- segments_tensors: (batch_size, max_seq_len_in_batch)
- masks_tensors   : (batch_size, max_seq_len_in_batch)
- label_ids       : (batch_size)
"""
```

```
# attention masks，將 tokens_tensors 裡頭不為 zero padding
# 的位置設為 1 讓 BERT 只關注這些位置的 tokens
masks_tensors = torch.zeros(tokens_tensors.shape,
                            dtype=torch.long)
masks_tensors = masks_tensors.masked_fill(
    tokens_tensors != 0, 1)
```

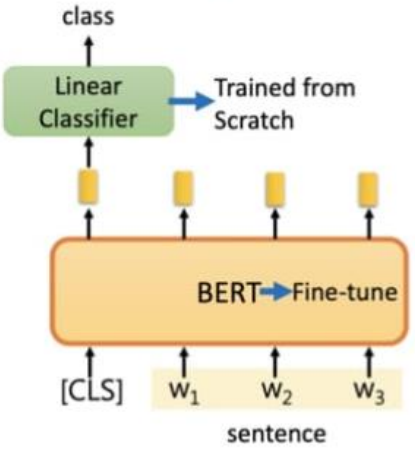
```
# 初始化一個每次回傳 64 個訓練樣本的 DataLoader
# 利用 `collate_fn` 將 list of samples 合併成一個 mini-batch 是關鍵
BATCH_SIZE = 64
trainloader = DataLoader(trainset, batch_size=BATCH_SIZE,
                         collate_fn=create_mini_batch)
```

Implements

LeeMeng-進擊的BERT (Pytorch)

單一句子分類任務

bertForSequenceClassification



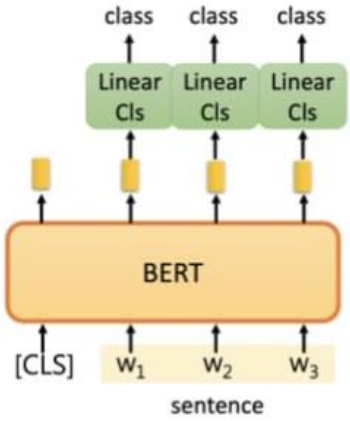
Input: single sentence, output: class

Example:
Sentiment analysis (our HW),
Document Classification

常見任務： SST-2, CoLA

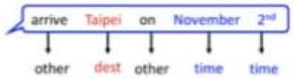
單一句子標註任務

bertForTokenClassification



Input: single sentence, output: class of each word

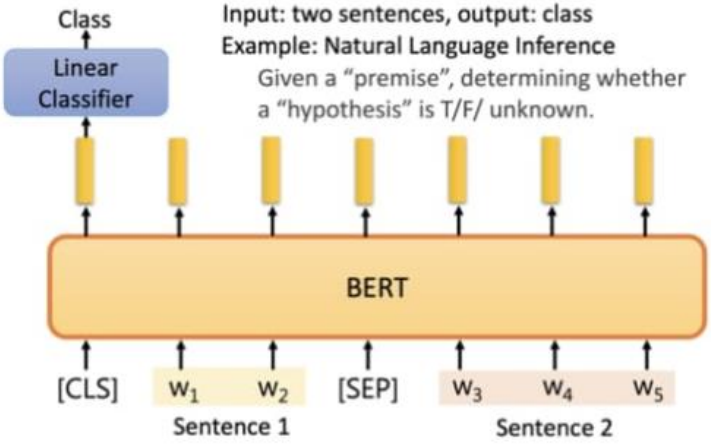
Example: Slot filling



常見任務： CoNLL-2003 NER

成對句子分類任務

bertForSequenceClassification



Input: two sentences, output: class

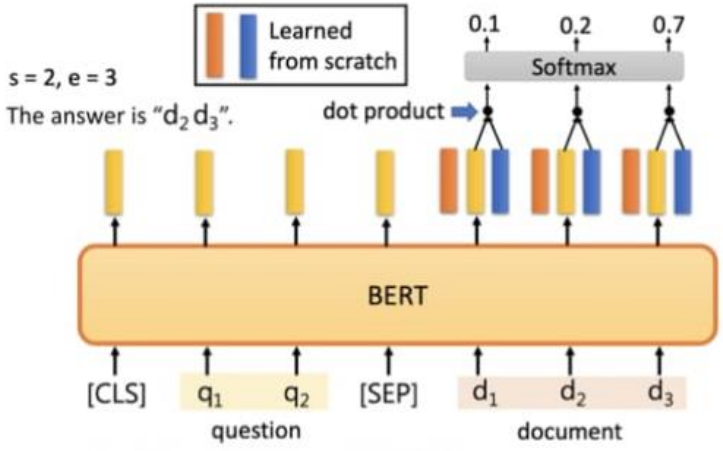
Example: Natural Language Inference

Given a "premise", determining whether a "hypothesis" is T/F/ unknown.

常見任務： MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

問答任務

bertForQuestionAnswering



常見任務： SQuAD v1.1, SQuAD 2.0

Implements

LeeMeng-進擊的BERT (Pytorch)

```
# 載入一個可以做中文多分類任務的模型，n_class = 3
GITHUB_REPO = "huggingface/pytorch-pretrained-BERT"
PRETRAINED_MODEL_NAME = "bert-base-chinese"
FINETUNE_TASK = "bertForSequenceClassification"

NUM_LABELS = 3

model = torch.hub.load(GITHUB_REPO,
                       FINETUNE_TASK,
                       PRETRAINED_MODEL_NAME,
                       num_labels=NUM_LABELS)
```

```
def forward(self, input_ids, token_type_ids=None, attention_mask=None, labels=None, ...):
    # BERT 輸入就是 tokens, segments, masks
    outputs = self.bert(input_ids, token_type_ids, attention_mask, ...)
    ...
    pooled_output = self.dropout(pooled_output)
    # 線性分類器將 dropout 後的 BERT repr. 轉成類別 logits
    logits = self.classifier(pooled_output)

    # 輸入有 labels 的話直接計算 Cross Entropy 回傳，方便！
    if labels is not None:
        loss_fct = CrossEntropyLoss()
        loss = loss_fct(logits.view(-1, self.num_labels), labels.view(-1))
        return loss
    # 回傳各類別的 logits
    return logits
```

5 Conclusion

References

BERT http://bit.ly/BERTpaper

語言模型發展 http://bit.ly/nGram2NNLM

語言模型預訓練方法 http://bit.ly/ELMo_OpenAIGPT_BERT

Attention Is All You Need http://bit.ly/AttIsAllUNeed

李弘毅-Transformer(Youtube) http://bit.ly/HungYiLee_Transformer

Illustrated Transformer http://bit.ly/illustratedTransformer

詳解Transformer http://bit.ly/explainTransformer

github/codertimo - BERT(pytorch) http://bit.ly/BERT_pytorch

Pytorch.org _ BERT http://bit.ly/pytorchorgBERT

實作假新聞分類 http://bit.ly/implementpaircls