DistilBERT Survey

DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter

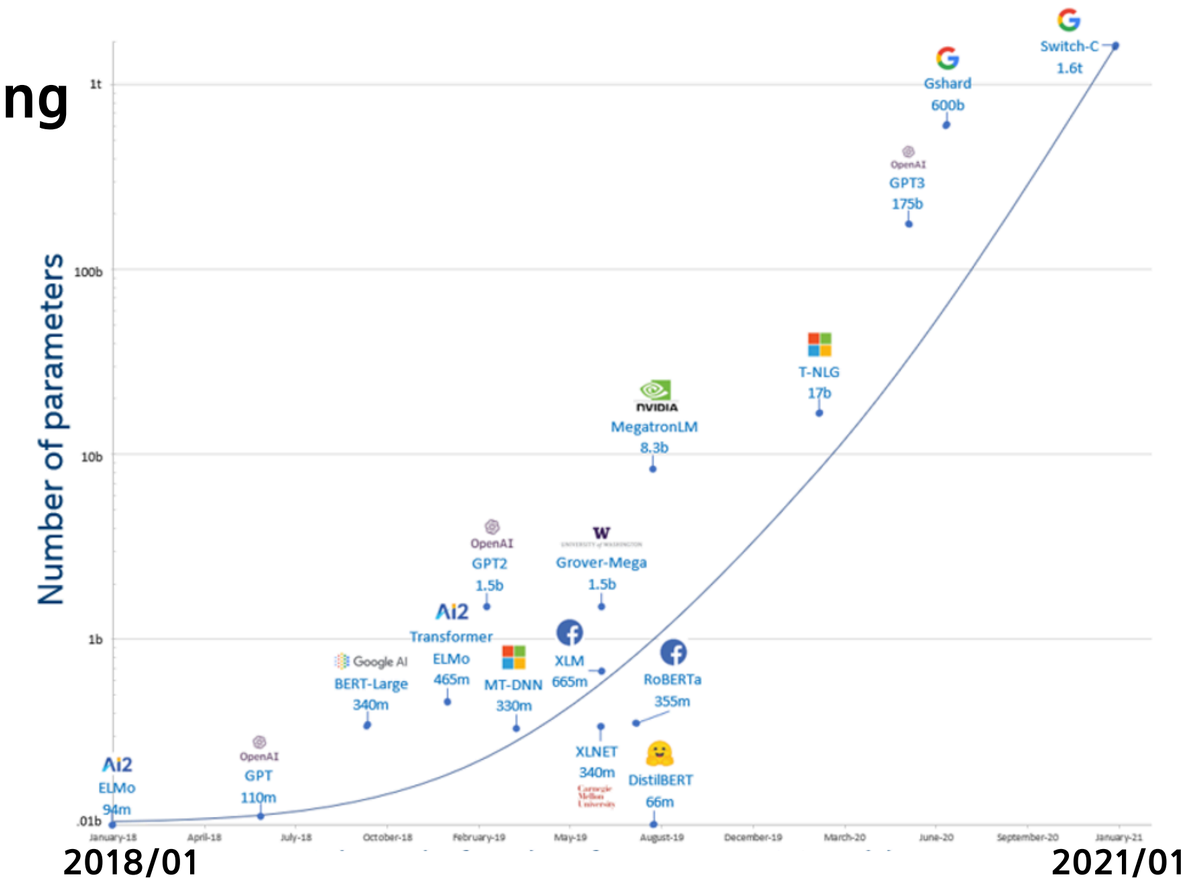
Written by Kim EunHo

Introduction

Pre-trained LM → Transfer learning

Large Scale LM disadvantage

- Expensive computatuinal cost
- Environmental cost



Parameter counts of pre-trained models

Written by Kim EunHo

출처 : DistilBERT, a distilled version of BERT smaller, faster, cheaper and lighter
https://developers-kr.googleblog.com/2020/06/how-hugging-face-achieved-2x-performance-boost-question-answering.html

DistilBERT

DistilBERT

- Parameter 40% lighter, Inf. time 60% faster
- Less Computation cost, Environmental cost
- Mobile runable.

→ Knowledge Distillation(Temperature + Triple loss)

Written by Kim EunHo

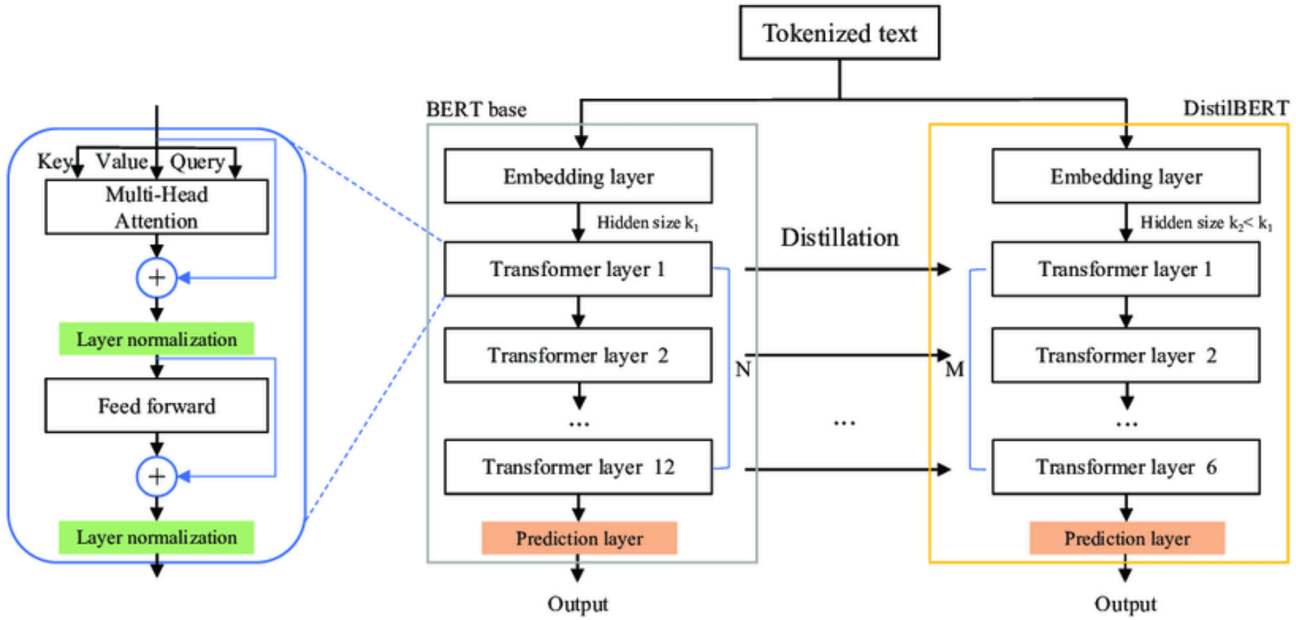
DistilBERT Architecture

Reduce the number of layers

BERT-base(12) → DistilBERT(6)

Difference

- Remove Token-type Embeddings
- Remove Pooler Layer

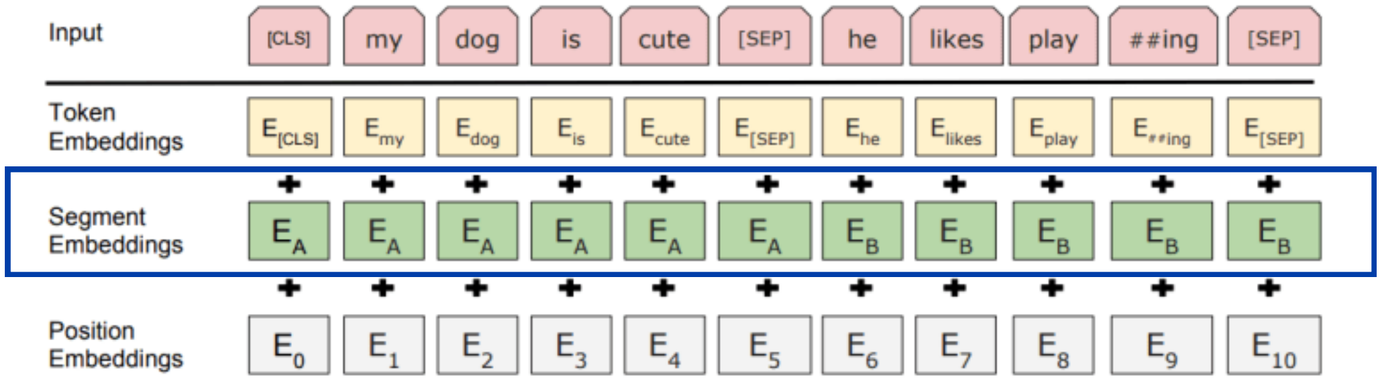


Written by Kim EunHo

출처 : https://www.researchgate.net/figure/The-DistilBERT-model-architecture-and-components_fig2_358239462

DistilBERT Architecture

Remove Token-type Embeddings



⇦ **Token-type Embeddings**

→ Do not perform Next Sentence Prediction(NSP)

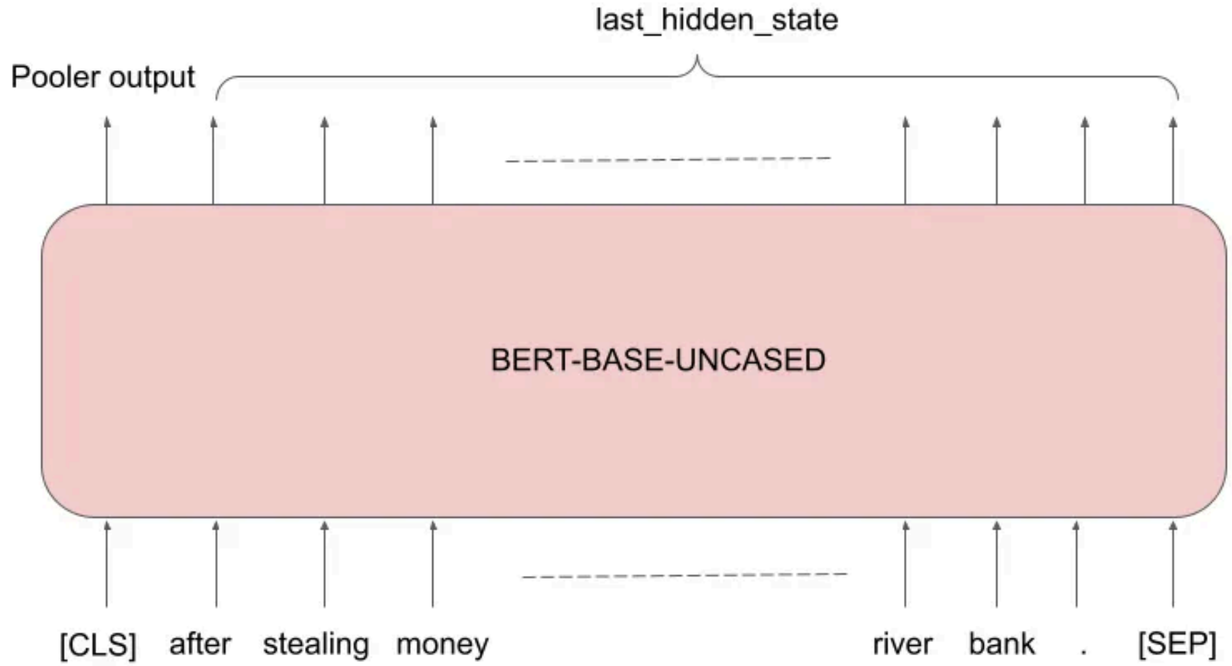
Written by Kim EunHo

출처 : https://www.researchgate.net/figure/BERT-input-representation-modified-image-from-8_fig3_345174555

DistilBERT Architecture

Remove Pooler Layer

- BERT calls Pooler Layer Class
- DistilBERT includes Pooler Layer



→ By Not calling class, simplified the model, reducing both training and inference times.

Written by Kim EunHo

출처 : https://niteshkumardew11.medium.com/implementing-bert-for-question-and-answer-67ccd0e8aae5

DistilBERT Architecture

BERT calls Pooler Layer

```python
class TFBertPooler(keras.layers.Layer):
    def __init__(self, config: BertConfig, **kwargs):
        super().__init__(**kwargs)

        self.dense = keras.layers.Dense(
            units=config.hidden_size,
            kernel_initializer=get_initializer(config.initializer_range),
            activation="tanh",
            name="dense",
        )
        self.config = config

    def call(self, hidden_states: tf.Tensor) -> tf.Tensor:
        first_token_tensor = hidden_states[:, 0]
        pooled_output = self.dense(inputs=first_token_tensor)

        return pooled_output
```

- Define the Dense to be used in the Pooler Layer.
- Use tanh as activation.

- Store the first token(CLS output) of "hidden_states" at first_token_tensor.
- Pass the first_token_tensor through the defined dense layer.
- Store the final output in “pooled_output” and return it.

Written by Kim EunHo

DistilBERT Architecture

DistilBERT includes Pooler Layer(When do Fine-Tuning Classification)

```
hidden_state = distilbert_output[0]
pooled_output = hidden_state[:, 0]
pooled_output = self.pre_classifier(pooled_output)
pooled_output = self.dropout(pooled_output, training=training)
logits = self.classifier(pooled_output)
```

- Store the first embedding of DistilBert in "hidden_state", and store the first token of "hidden_state" in "pooled_output".
- Pass “pooled_output” through “pre_classifier” and do pre-processing, classification.

```
self.pre_classifier = tf.keras.layers.Dense(
    config.dim,
    kernel_initializer=get_initializer(config.initializer_range),
    activation="relu",
    name="pre_classifier",
)
```

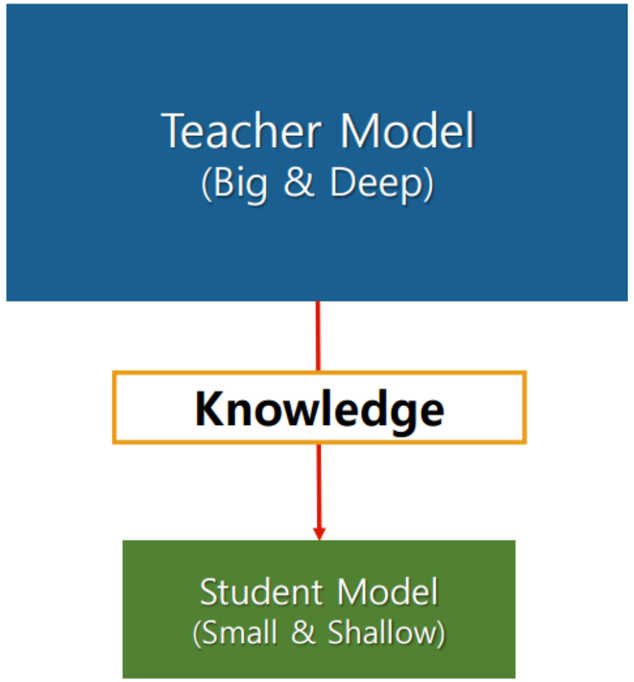
- Define “pre_classifier”.
- This dense layer performs preprocessing.

Written by Kim EunHo

Knowledge Distillation

Knowledge Distillation

- Knowledge Distillation = Compression Technique which Knowledge from large-scale Model to transfer to a small Model
- Teacher Model = Large-scale Model
- Student Model = Small Model
- Softmax + Temperature
- Use Triple Loss
  - Distillation Loss
  - Masked Language Modeling Loss
  - Cosine Embedding Loss



Written by Kim EunHo

출처 : https://velog.io/@qtly_u/%EB%AA%A8%EB%8D%B8-%EA%B2%BD%EB%9F%89%ED%99%94-%EA%B8%B0%EB%B2%95-Knowledge-Distillation

Knowledge Distillation

Why use Temperature?

- Randomness
- To transfer the wealth of knowledge
- Temperature makes probability smoothly

softmax

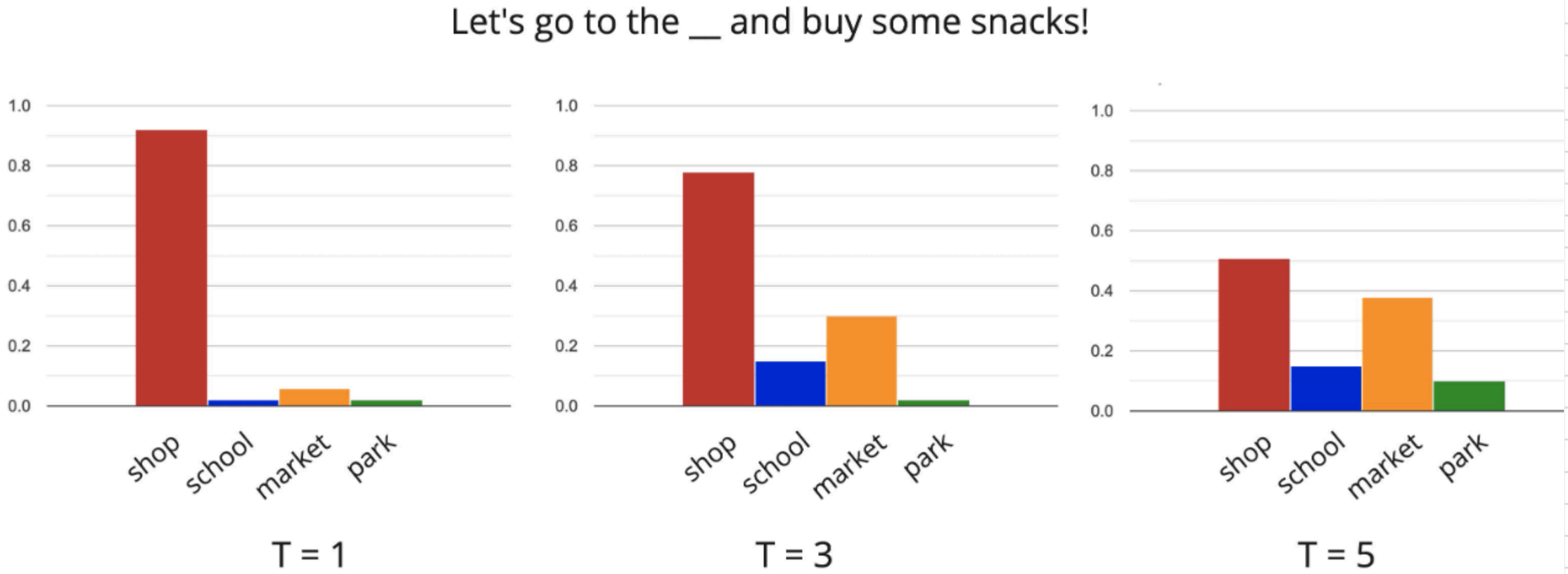
$$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} + T = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

softmax temperature

Written by Kim EunHo

출처 : https://www.youtube.com/watch?app=desktop&v=ATHwSCg_W4A&ab_channel=DataScienceinyourpocket

Knowledge Distillation

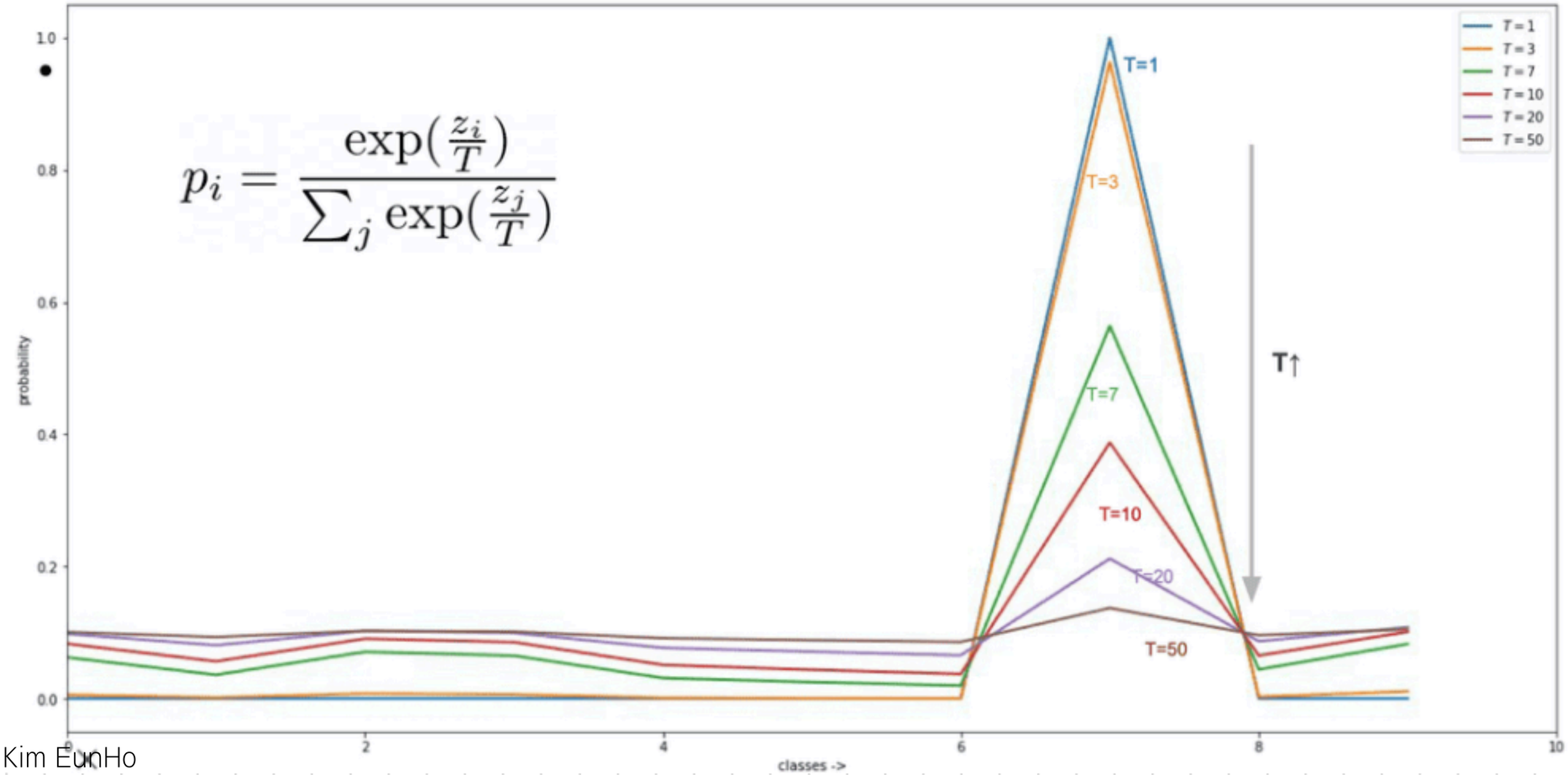


Written by Kim EunHo

출처 : https://alexnim.com/coding-projects-knowledge-distillation.html

Knowledge Distillation

$$p_i = \frac{\exp(\frac{z_i}{T})}{\sum_j \exp(\frac{z_j}{T})}$$



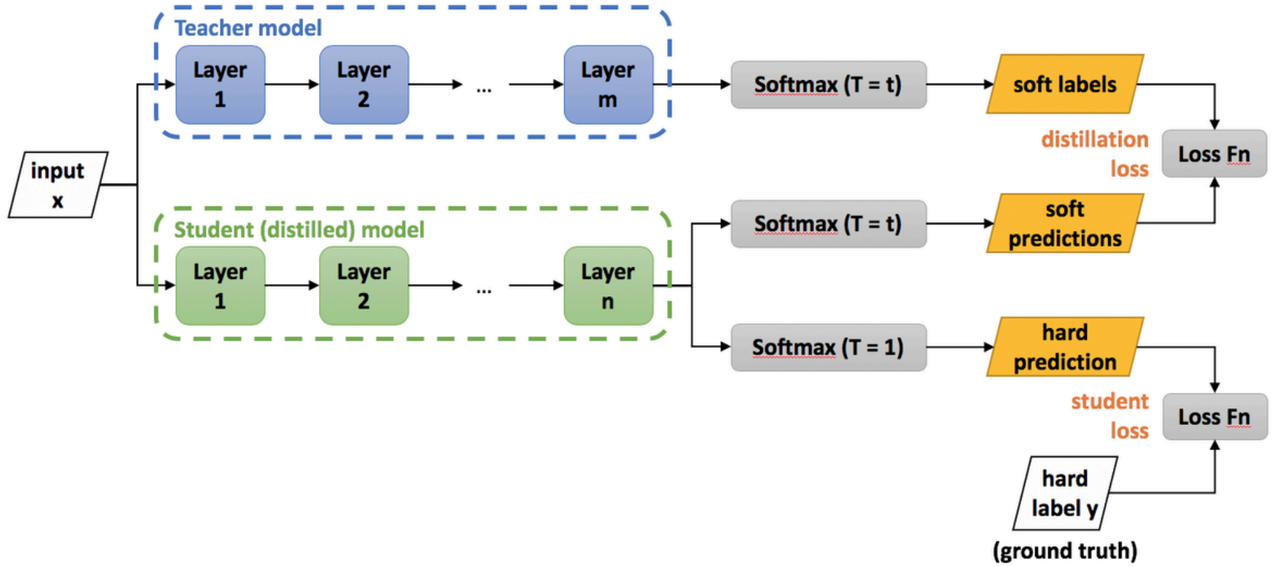
Written by Kim EunHo

출처 : https://velog.io/@dldydldy75/%EC%A7%80%EC%8B%9D-%EC%A6%9D%EB%A5%98-Knowledge-Distillation

Knowledge Distillation

Basic Knowledge Distillation Architecture

1. Pre-Train Teacher
2. Get soft labels generated by the Teacher
3. Train Student Model with same dataset as the Teacher
4. Get soft labels generated by the Student
5. Get hard labels generated by the Student



Written by Kim EunHo

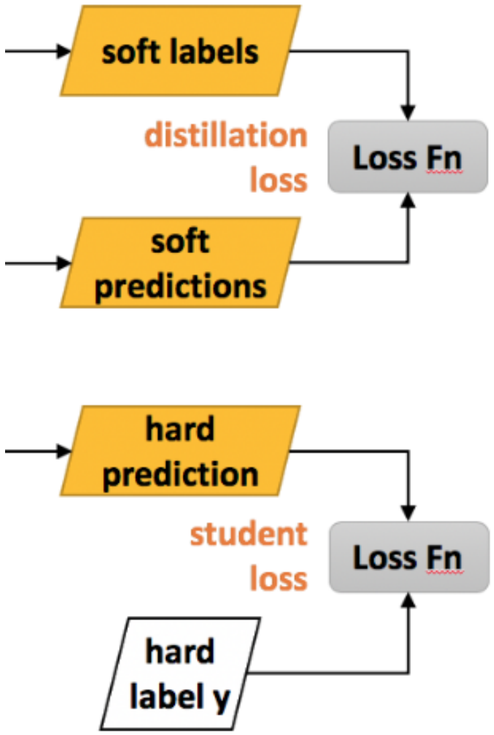
출처 : https://light-tree.tistory.com/196

Knowledge Distillation

Why use Distillation loss, Student loss?

- Distillation loss → Learning the characteristics of the teacher model

- Student loss → Predict correct answer



Written by Kim EunHo

출처 : https://light-tree.tistory.com/196

Knowledge Distillation

Knowledge Distillation Architecture of DistilBERT

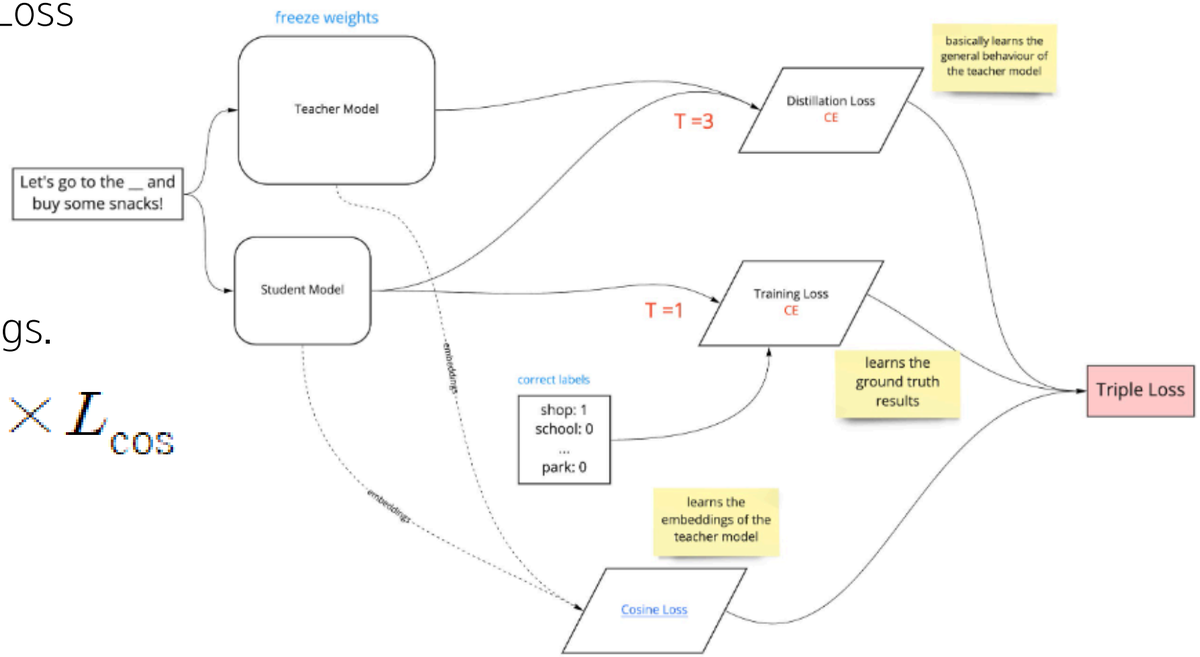
- Distillation Loss
- Student Loss → Masked Language Modeling Loss
- Cosine Embedding Loss(New Loss)
    - Add cosine similarity loss between

      Teacher's last hidden state embeddings

      and Student's last hidden state embeddings.

$$L_{triple} = \alpha \times L_{distil} + \beta \times L_{MLM} + \gamma \times L_{\cos}$$

$$\alpha + \beta + \gamma = 1$$



Written by Kim EunHo

출처 : https://towardsdatascience.com/distilbert-11c8810d29fc

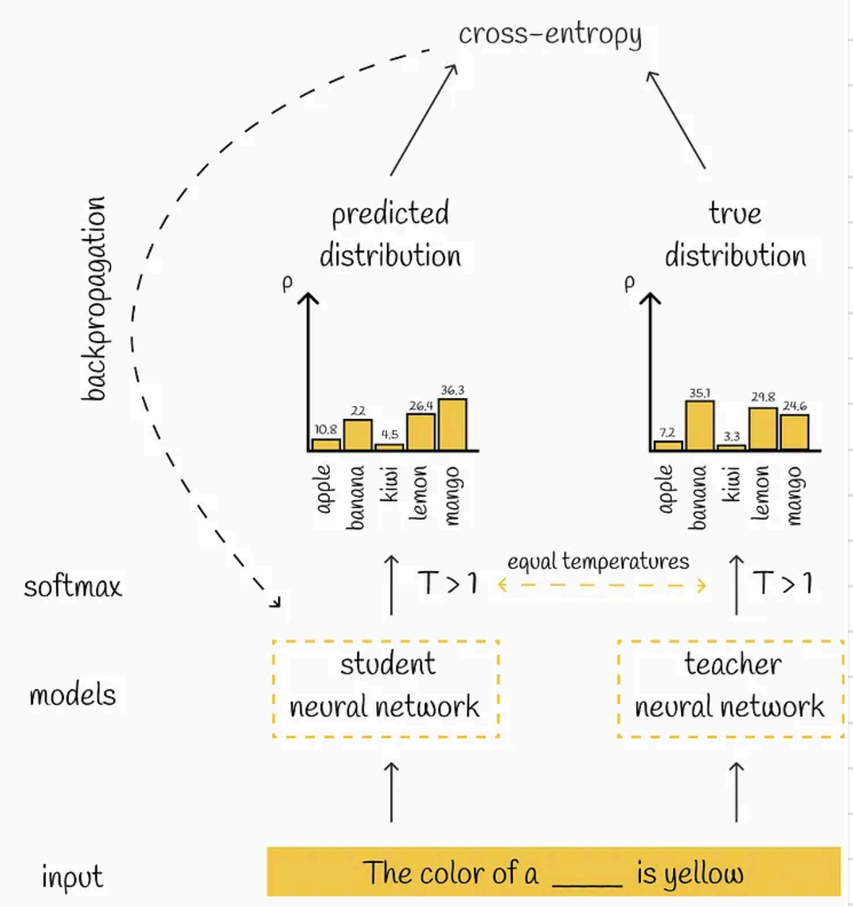
Knowledge Distillation

Why use Triple Loss??

- Distillation Loss
  - Goal : Minimising the loss about the Teacher's soft labels
  - Action : To align the Student's output to Teacher's output

$$L_{distil} = (\sigma(\frac{Z_s}{T}), \sigma(\frac{Z_t}{T}))$$

σ = softmax
T = temperature(same)
Z = logits(outputs)



Written by Kim EunHo

출처 : https://towardsdatascience.com/distilbert-11c8810d29fc, https://velog.io/@tjdcjffff/Distilling-the-Knowledge-in-a-Neural-Network

Knowledge Distillation

Distillation Loss

```
self.ce_loss_fct = nn.KLDivLoss(reduction="batchmean")

loss_ce = (
    self.ce_loss_fct(
        nn.functional.log_softmax(s_logits_slct / self.temperature, dim=-1),
        nn.functional.softmax(t_logits_slct / self.temperature, dim=-1),
    )
    * (self.temperature) ** 2
)
loss = self.alpha_ce * loss_ce
```

Apply softmax to each Model's output divided by T and get Loss by CE

Multiply “loss_ce” and “alpha_ce”(Ratio) and store it in “loss”

Written by Kim EunHo

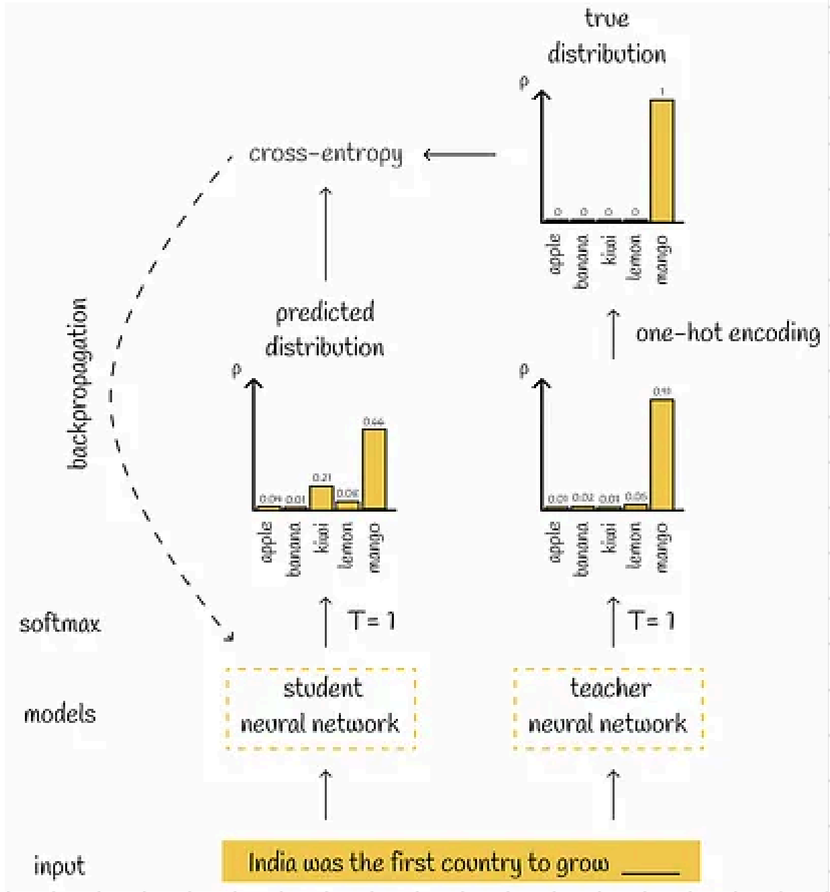
Knowledge Distillation

Why use Triple Loss??

- Masked Language Modeling Loss
  - Goal : Pursuit Cross Entropy between the hard prediction and hard label
  - Action : Predict masked language

$$L_{MLM} = (\sigma(Z_s), \hat{y})$$

$$\hat{y} = hard\ labels$$



Written by Kim EunHo

출처 : https://towardsdatascience.com/distilbert-11c8810d29fc, https://velog.io/@tjdcjffff/Distilling-the-Knowledge-in-a-Neural-Network

Knowledge Distillation

Masked Language Modeling Loss

```
self.lm_loss_fct = nn.CrossEntropyLoss(ignore_index=-100)

if self.alpha_mlm > 0.0:
    loss_mlm = self.lm_loss_fct(s_logits.view(-1, s_logits.size(-1)), lm_labels.view(-1))

    loss += self.alpha_mlm * loss_mlm
```

Add to “loss” by multiplying “loss_mlm” and “alpha_mlm”(Ratio).

- Change “s_logits” to 2-dimensions.
- Change “lm_labels” to 1-dimensions.
- Get “loss_mlm” through CE

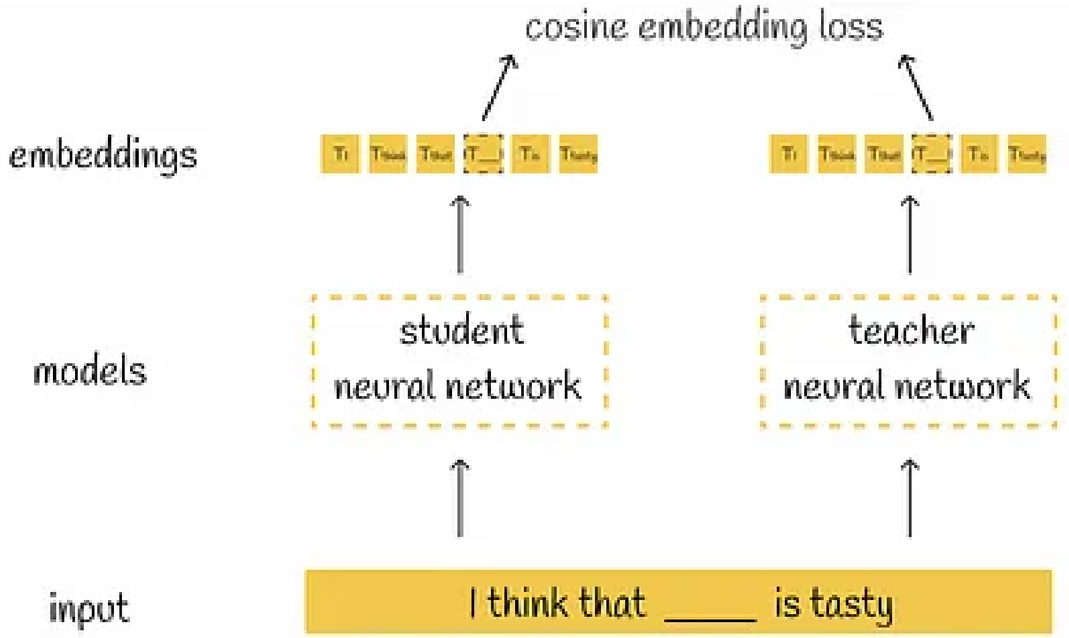
Written by Kim EunHo

Knowledge Distillation

Why use Triple Loss??

- Cosine Embedding Loss
  - Goal : Make cosine similarity of the Student's last hidden state embeddings and Teacher's last hidden state embeddings is 1
  - Action : To align the Student's last hidden state embeddings to Teacher's last hidden state embeddings.

$$L_{cos}(x, y) = 1 - cos(x, y)$$
$$-1 \leq cos(x, y) \leq 1$$
$$= 0 \leq 1 - cos(x, y) \leq 2$$



Written by Kim EunHo

출처 : https://towardsdatascience.com/distilbert-11c8810d29fc, https://velog.io/@tjdcjffff/Distilling-the-Knowledge-in-a-Neural-Network

Knowledge Distillation

Cosine Embedding Loss

```
if self.alpha_cos > 0.0:
    s_hidden_states = s_hidden_states[-1]
    t_hidden_states = t_hidden_states[-1]
    mask = attention_mask.unsqueeze(-1).expand_as(s_hidden_states)
    assert s_hidden_states.size() == t_hidden_states.size()
    dim = s_hidden_states.size(-1)

    s_hidden_states_slct = torch.masked_select(s_hidden_states, mask)
    s_hidden_states_slct = s_hidden_states_slct.view(-1, dim)
    t_hidden_states_slct = torch.masked_select(t_hidden_states, mask)
    t_hidden_states_slct = t_hidden_states_slct.view(-1, dim)

    target = s_hidden_states_slct.new(s_hidden_states_slct.size(0)).fill_(1)
    loss_cos = self.cosine_loss_fct(s_hidden_states_slct, t_hidden_states_slct, target)
    loss += self.alpha_cos * loss_cos
```

Save the last hidden state embeddings for each Model to each variables.

Only the index of masked words is extracted, stored in other variables, and dimensions are adjusted.

- Get similarity between “s_hidden_states_slct” and “t_hidden_states_slct”.
- Get loss between the similarity and target(=1).

Written by Kim EunHo

Knowledge Distillation

Triple Loss

```
loss = self.alpha_ce * loss_ce
loss += self.alpha_mlm * loss_mlm
loss += self.alpha_cos * loss_cos

self.optimize(loss)
```

- Keep adding for each loss to get Triple Loss.
- Optimize Triple Loss.

Written by Kim EunHo

Student Initialization

Teacher’s weight → Student’s weight initialization

- Same dimensions as Teacher model

Difference

- Teacher is initialized randomly
- Student is initialized with Teacher’s weight

→ Initialized with the weights of the learned Teacher

Written by Kim EunHo

Ablation Study

Ablation

1. Ablation Distillation Loss
2. Ablation Cosine Embedding Loss
3. Ablation Masked Language Modeling Loss
4. No Ablation + random weight initialization

Default : Use Triple Loss + Teacher's weight
GLUE = Average F1 score of 9 Tasks

→ Learn more from Teacher than Training Data

Ablation	Variation on GLUE macro-score
$\emptyset$ - L_{cos} - L_{mlm}	-2.96
L_{ce} - $\emptyset$ - L_{mlm}	-1.46
L_{ce} - L_{cos} - $\emptyset$	-0.31
Triple loss + random weights initialization	-3.69

Student's weight

Written by Kim EunHo

출처 : DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter

Performance Comparison

Model	**Score**	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
ELMo	68.7	44.1	68.6	76.6	71.1	86.2	53.4	91.5	70.4	56.3
BERT-base	79.5	56.3	86.7	88.6	91.8	89.6	69.3	92.7	89.0	53.5
DistilBERT	77.0	51.3	82.2	87.5	89.2	88.5	59.9	91.3	86.9	56.3

Performace comparison for each Task

Model	IMDb (acc.)	SQuAD (EM/F1)
BERT-base	93.46	81.2/88.5
DistilBERT	92.82	77.7/85.8
DistilBERT (D)	-	79.1/86.9

Performance comparison of Downstream Task

Model	# param. (Millions)	Inf. time (seconds)
ELMo	180	895
BERT-base	110	668
DistilBERT	66	410

Param / Inf. time for each Model

Written by Kim EunHo

출처 : DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter

Application Field

1. Multiple-choice
2. **Question-answering** ✓
3. Summarization
4. Text-classification
5. Token-classification
6. Translation

Written by Kim EunHo

Question-answering model

Use Hugging face's Question-answering model

```
11065/11065 [==============================] - 2611s 235ms/step - loss: 1.4423 - end_logits_accuracy: 0.6228 - start_logits_accuracy: 0.5842
Epoch 2/3
11065/11065 [==============================] - 2603s 235ms/step - loss: 0.8246 - end_logits_accuracy: 0.7698 - start_logits_accuracy: 0.7326
Epoch 3/3
11065/11065 [==============================] - 2610s 236ms/step - loss: 0.5055 - end_logits_accuracy: 0.8555 - start_logits_accuracy: 0.8251
```

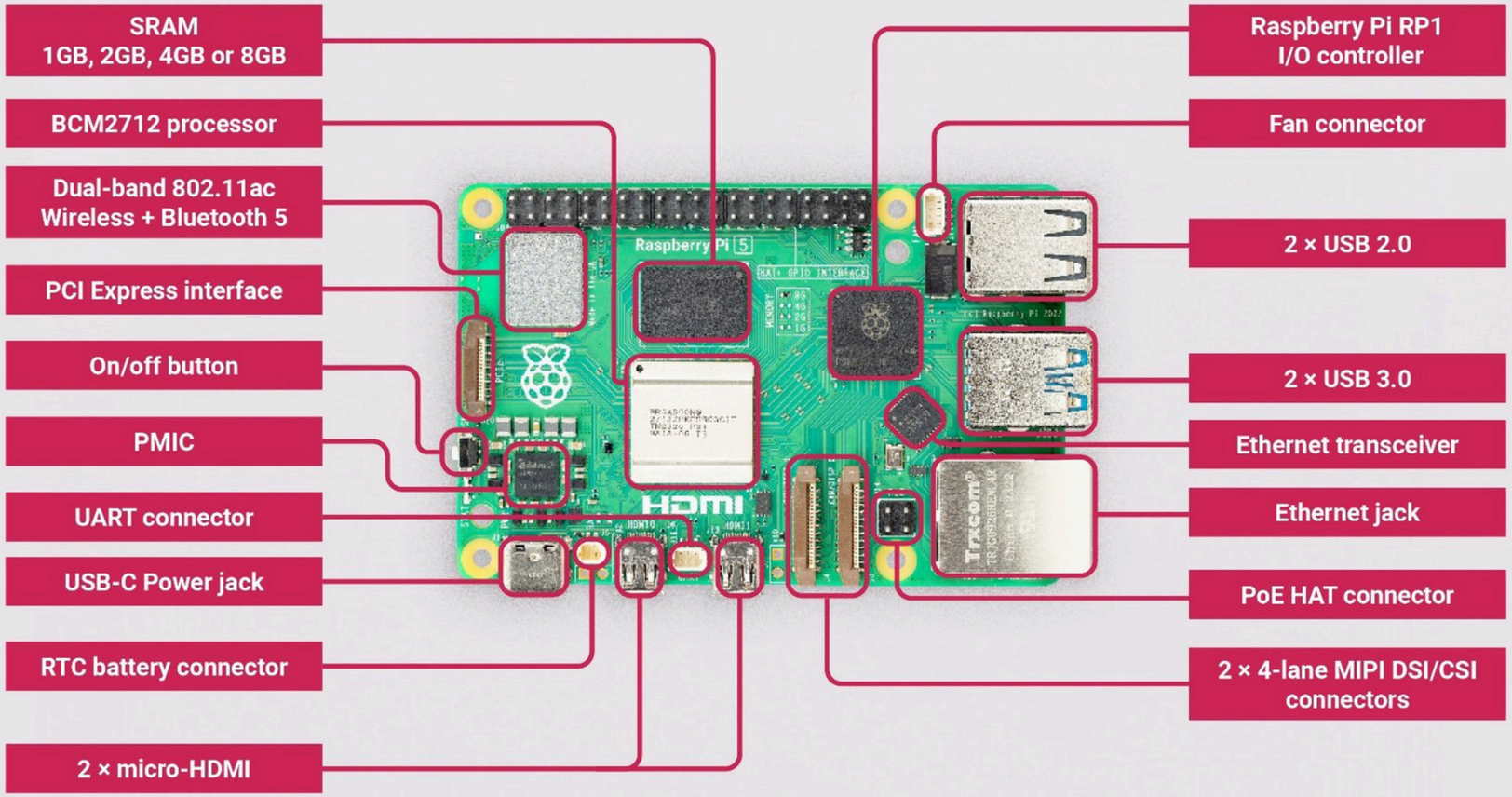
Make Q/A model to fine-tuning distilbert-base-uncased

all_results	2024-02-18 오전 6:10	JSON 파일
config	2024-02-18 오전 6:10	JSON 파일
eval_nbest_predictions	2024-02-18 오전 6:10	JSON 파일
eval_predictions	2024-02-18 오전 6:10	JSON 파일
tf_model.h5	2024-02-18 오전 6:10	H5 파일

This is result of fine-tuning and use tf_model for Question-answering

Written by Kim EunHo

Raspberry Pi 5



출처 : https://www.raspberrypi.com/documentation/computers/raspberry-pi-5.html

Raspberry Pi 4 vs 5

Raspberry Pi 4

- CPU: Broadcom BCM2711 Quad core Cortex-A72 64-bit 1.8GHz
- MEMORY: LPDDR4-3200 SDRAM

I/O

- HDMI: micro-HDMI 4Kp60 x2
- Audio: 4-pole stereo audio and video jack
- DIS: 2-lane MIPI DSI display connector
- CIS: 2-lane MIPI DSI camera connector
- PCIe: N/A
- SDcard: microSD card slot
- USD-C: DC 5V/3A
- on/off SW: N/A
- RTC: N/A
- Connector: N/A

Written by Kim EunHo

Raspberry Pi 5

- CPU: Broadcom BCM2712 Quad core Cortex-A76 64-bit 2.4GHz
- MEMORY: LPDDR4X-4267 SDRAM

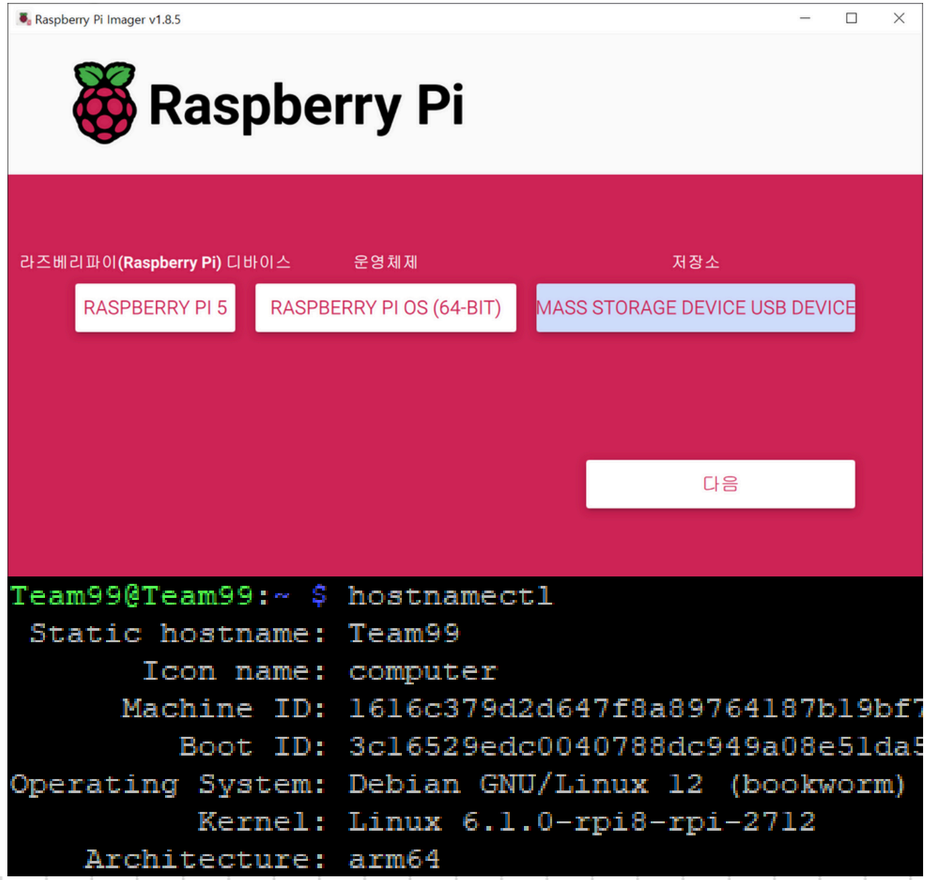
I/O

- HDMI: micro-HDMI 4Kp60(HDR supported) x2
- Audio:removed
- DIS/CIS: 4-lane MIPI camera/display connector x2
- PCIe: PCIe 2.0 interface (Exclusive HAT/ adapter required)
- SDcard: microSD card slot, SDR104 Supports high-speed mode
- USD-C: DC 5V/5A
- on/off SW: on/off power switch
- RTC: Real-time clock, External battery connector
- Connector: UART Connector / FAN power only connector

Raspberry Pi 5 Environment

Environment

- Device : RASPBERRY PI 5
- OS : Raspberry Pi OS(64bit)
  - Linux Based
  - Debian GNU/Linux 12 (bookworm)
- Architecture : ARM64



Written by Kim EunHo

Required Libraries

1. Tensorflow

2. Transformers

3. Datasets

Written by Kim EunHo

PC “run_qa”

```
1   import tensorflow as tf
2   from transformers import DistilBertTokenizer, TFDistilBertForQuestionAnswering
3
4   model_h5_path = "C:/transformers/examples/tensorflow/question-answering/result/tf_model.h5"
5   config_json_path = "C:/transformers/examples/tensorflow/question-answering/result/config.json"
6
7   loaded_model = TFDistilBertForQuestionAnswering.from_pretrained(model_h5_path, config=config_json_path)
8
9   tokenizer = DistilBertTokenizer.from_pretrained("distilbert-base-uncased")
10
11  context = input("Enter the Original Sentence: ")
12  question = input("Enter the Question: ")
```

1 ~ 2 : Import Library

4 ~ 5 : Load Model & Config

7 : Build model with pretrained model & config

9 : Load Tokenizer

11 ~ 12 : Set context & question manually

Written by Kim EunHo

PC “run_qa”

```
14    inputs = tokenizer(question, context, return_tensors="tf", padding=True, truncation=True)
15
16    outputs = loaded_model(inputs)
17    start_logits, end_logits = outputs.start_logits, outputs.end_logits
18
19    start_index = tf.argmax(start_logits, axis=1).numpy()[0]
20    end_index = tf.argmax(end_logits, axis=1).numpy()[0]
```

14：Do Tokenize

- **question, context：Need information to answer.**
- **return_tensors：Define type when return. It follows used framework.**
- **padding：Adjust the length of the input tokens.**
- **truncation：Limit the max length of input. If exceed the limit, throw it away as much as exceed it.**

16：Pass “inputs” through model

17：Get start/end token of outputs that has highest possibility about start/end token.

19 ~ 20：Get highest value’s Index, transform to numpy array and select first token of array.

Written by Kim EunHo

PC “run_qa”

```
21
22    all_tokens = tokenizer.convert_ids_to_tokens(inputs.input_ids.numpy()[0])
23    answer = tokenizer.convert_tokens_to_string(all_tokens[start_index:end_index + 1])
24
25    print("Answer:", answer)
```

22 : Convert Token IDs to numpy array and convert it to Token.

 # [[101, 363, 258, 5425, …]] -> [101, 363, 258, 5425, …] -> [’[CLS]’, ‘the’, ‘capital’, ‘of’, …]’]

23 : Select Tokens to use answer and convert to string.

 # all_tokens = [’[CLS]’, ‘the’, ‘capital’, ‘of’, …]’]

 # all_tokens[start_index:end_index + 1] = [’seoul’]

25 : Print “answer”

Written by Kim EunHo

PC “run_qa” Result

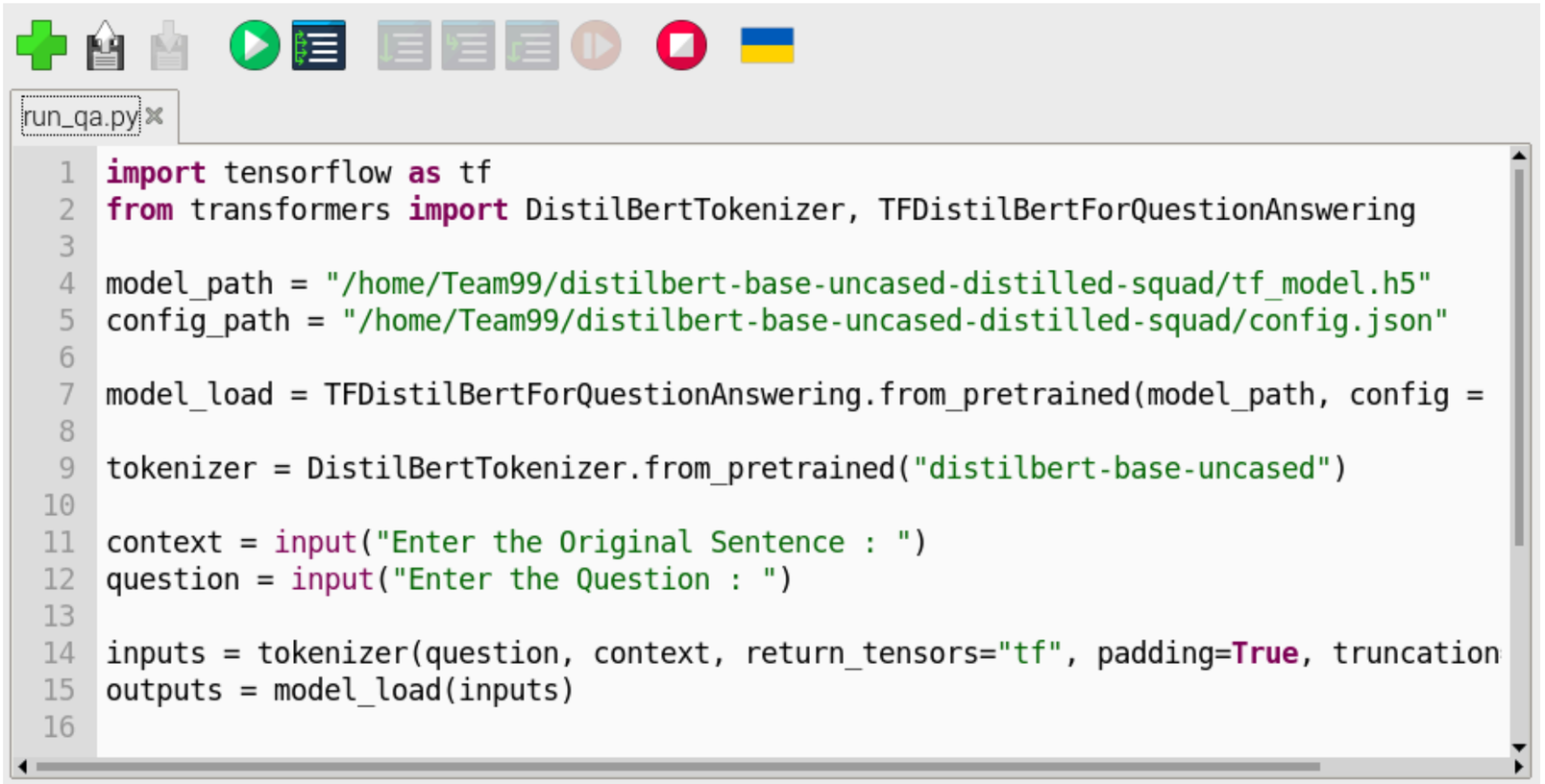
```
Enter the Original Sentence: The capital of Korea is Seoul
Enter the Question: Where is the capital of Korea?
Answer: seoul
```

→ Result of run_qa.py

Written by Kim EunHo

Raspberry pi 5 “run_qa.py”

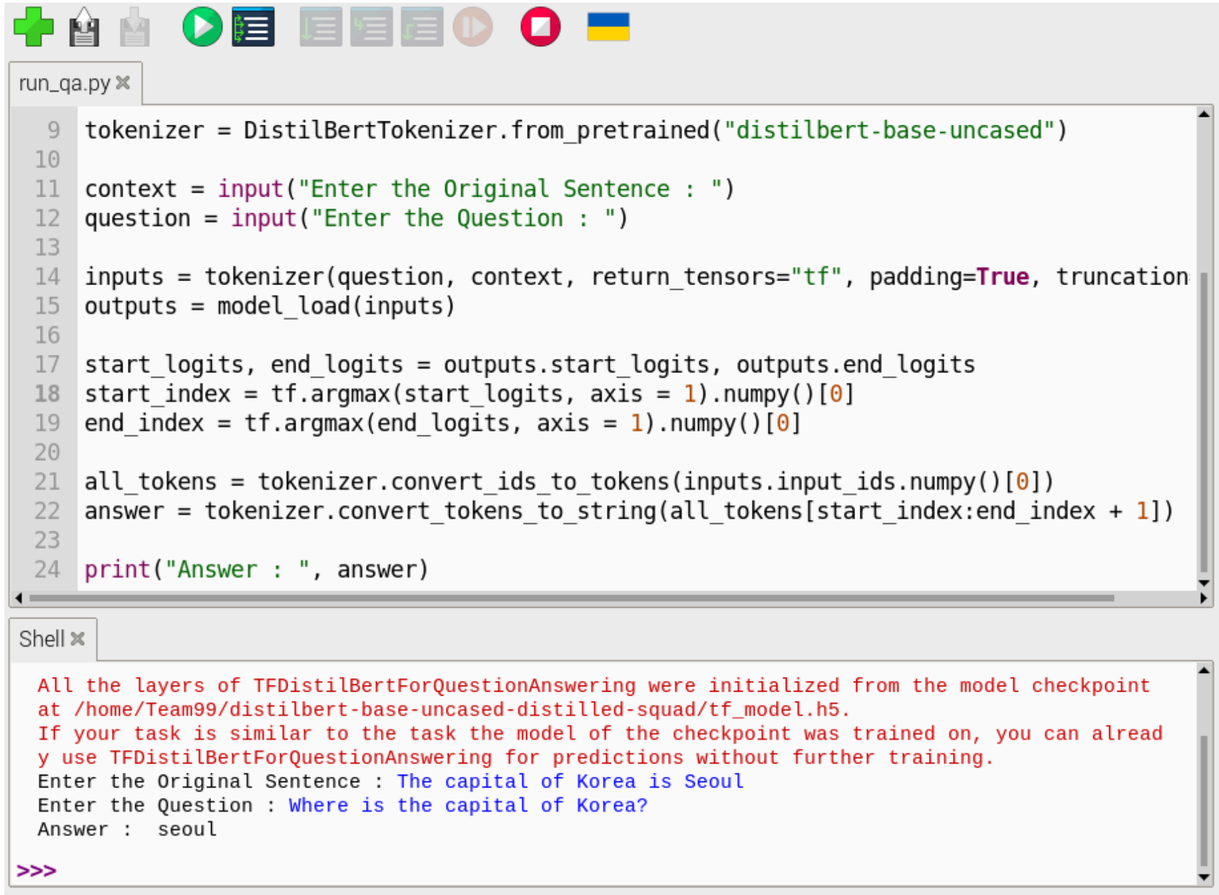
Single - context



Written by Kim EunHo

Raspberry pi 5 “run_qa.py”

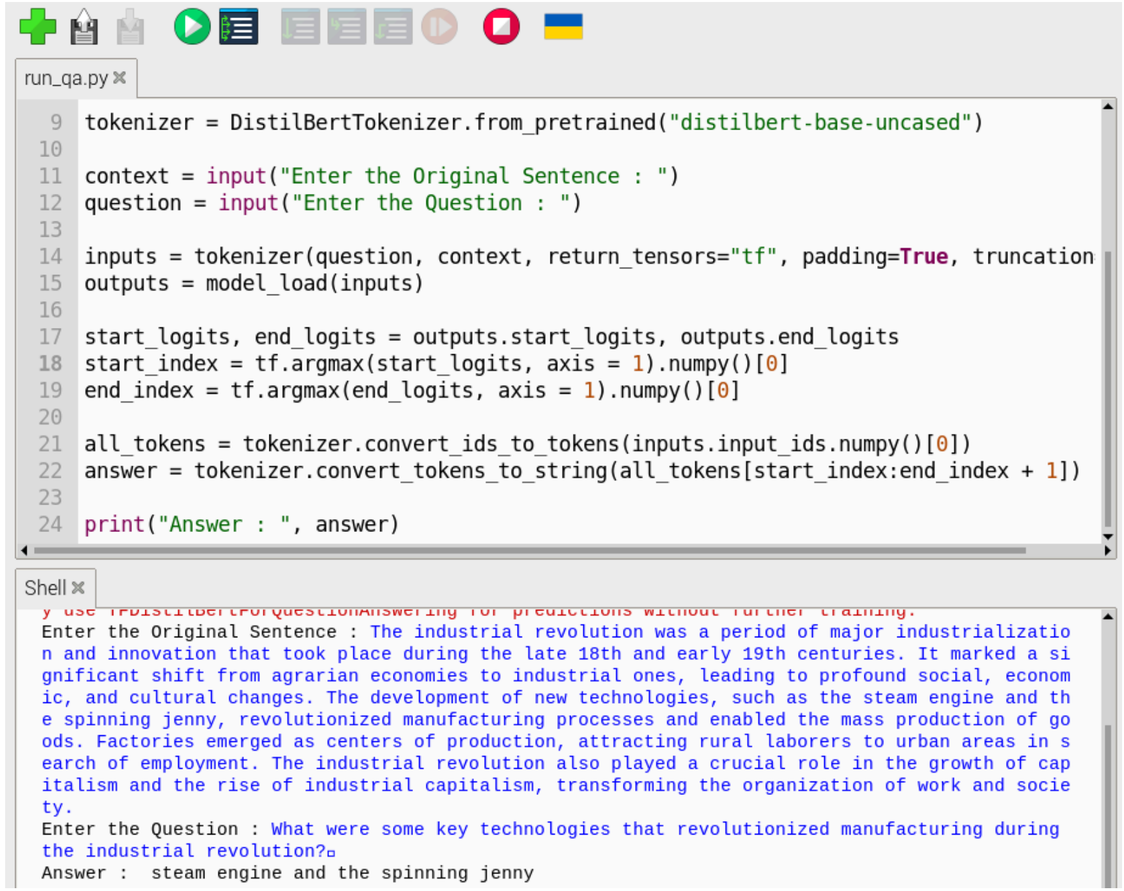
Single - context



Written by Kim EunHo

Raspberry pi 5 “run_qa.py”

Multi - context



Written by Kim EunHo

Performance Measurement Code

```
20    def calculate_f1_score(tokenizer, model, contexts, questions, answers):
21        f1_scores = []
22        em_scores = []
23        total_time = 0.0
24        for context, question, answer in zip(contexts, questions, answers):
25            inputs = tokenizer(question, context, return_tensors="tf", padding=True, truncation=False)
26            start_time = time.time()
27            outputs = model(inputs)
28            end_time = time.time()
29            start_logits, end_logits = outputs.start_logits, outputs.end_logits
30
31            start_index = tf.argmax(start_logits, axis=1).numpy()[0]
32            end_index = tf.argmax(end_logits, axis=1).numpy()[0]
```

20: Defines a function to calculate EM/F1 Score.

21 ~ 23: Define a list to store EM / F1 Score and Initialize total_time.

24 ~ 25: It receives parameters, proceeds with tokenization, and stores them in inputs.

26 ~ 28: Get total Inference time

29: Get start/end token of outputs that has highest possibility about start/end token.

31 ~ 32:Get highest value's Index, transform to numpy array and select first token of array.

Written by Kim EunHo

Performance Measurement Code

```
34          predicted_answer = tokenizer.decode(inputs["input_ids"][0][start_index:end_index+1])
35
36          f1_score = calculate_single_f1_score(predicted_answer, answer)
37          f1_scores.append(f1_score)
38
39          em_score = 1 if predicted_answer.strip().lower() == answer.strip().lower() else 0
40          em_scores.append(em_score)
41
42          total_time += (end_time - start_time)
43
44      return np.mean(f1_scores) * 100, em_scores, total_time
```

34 : Gets the token ID list of the input sentence, extracts the token ID portion corresponding to the answer, and decodes it to the original text.

36 ~ 37 : Use the function to get the f1 score value and create a list using the append method

39 ~ 40 : Give em_score 1 if remove spaces and lowercase strings match, or 0 if not.

42 : Get total Inference time.

44 : Return f1 scores(f1 score's list), em_scores(em_score's list,) total time(Inf. time).

Written by Kim EunHo

Performance Measurement Code

```
46    def calculate_single_f1_score(prediction, true_answer):
47        prediction_tokens = prediction.lower().split()
48        true_tokens = true_answer.lower().split()
49
50        common_tokens = set(prediction_tokens) & set(true_tokens)
51
52        precision = len(common_tokens) / len(prediction_tokens) if len(prediction_tokens) > 0 else 0
53        recall = len(common_tokens) / len(true_tokens) if len(true_tokens) > 0 else 0
54
55        f1_score = (2 * precision * recall) / (precision + recall) if (precision + recall) > 0 else 0
56
57        return f1_score
```

47 ~ 48 : Split text into word units and lowercase

50 : Find common_tokens and store them

52 ~ 53 : Get precision and recall as length of common_tokens, prediction_tokens and true_tokens

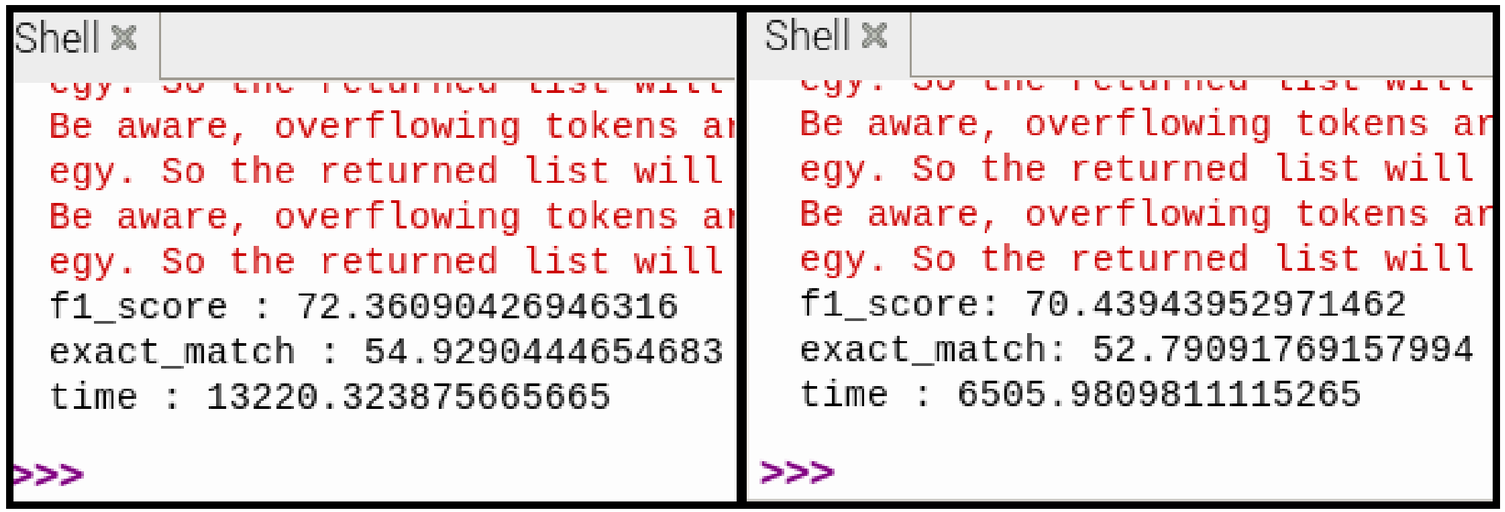
55 : Get f1_score as precision and recall

57 : Return f1_score

Written by Kim EunHo

Performance Measurement

EM(Exact Match) / F1 / Inference Time



(a)BERT-base (b)DistilBERT

Written by Kim EunHo

참고 문헌

- DistilBERT, a distilled version of BERT smaller, faster, cheaper and lighter
- https://zerojsh00.github.io/posts/DistilBERT/
- https://velog.io/@dldydldy75/%EC%A7%80%EC%8B%9D-%EC%A6%9D%EB%A5%98-Knowledge-Distillation
- https://www.researchgate.net/figure/The-DistilBERT-model-architecture-and-components_fig2_358239462
- https://www.researchgate.net/figure/BERT-input-representation-modified-image-from-8_fig3_345174555
- https://towardsdatascience.com/distilbert-11c8810d29fc, https://velog.io/@tjdcjffff/Distilling-the-Knowledge-in-a-Neural-Network
- https://light-tree.tistory.com/196
- https://alexnim.com/coding-projects-knowledge-distillation.html
- https://www.raspberrypi.com/documentation/computers/raspberry-pi-5.html

Written by Kim EunHo

Q/A

Written by Kim EunHo