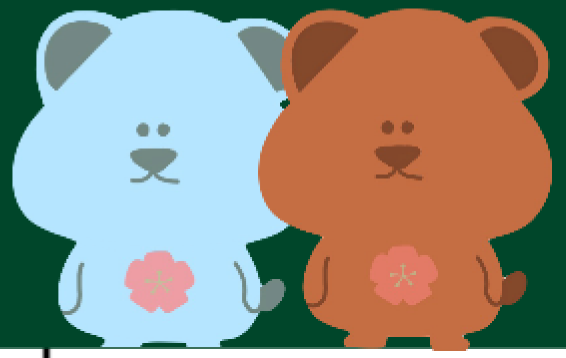


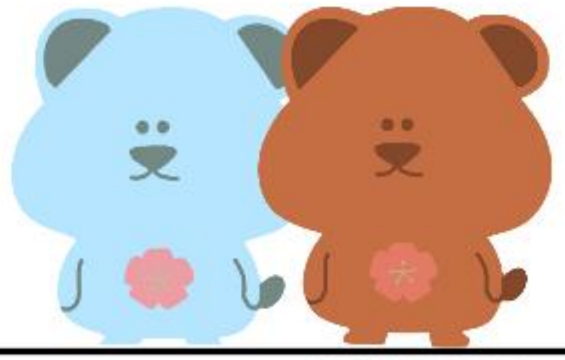
고급심화 _ NLP 뽀개기

NLP 뽀개기

배수현(5기) 정채윤(5기) 황채원(2기)

EWHA EURON

목차



EWHA EURON

#01 대회 소개: Detect AI Generated Text

THE LEARNING AGENCY LAB · FEATURED CODE COMPETITION · A MONTH AGO

Late Submission ···

LLM - Detect AI Generated Text

Identify which essay was written by a large language model

목표

LLM이 작성한 에세이와 중고등학교 학생이 작성한 에세이를 분류하는 모델 개발

제공된 데이터

id	prompt_id	text	generated
1378 unique values	0 1	1378 unique values	0 1
0059830c	0	Cars. Cars have been around since they became famous in the 1900s, when Henry Ford created and built...	0

train_essay.csv

prompt_id	prompt_name	instructions	source_text
2 total values	2 unique values	2 unique values	2 unique values
0	Car-free cities	Write an explanatory essay to inform fellow citizens about the advantages of limiting car usage. You...	# In German Suburb, Life Goes On Without Cars by Elisabeth Rosenthal 1 VAUBAN, Germany-Residents of...
1	Does the electoral college work?	Write a letter to your state senator in which you argue in favor of keeping the Electoral College or...	# What Is the Electoral College? by the Office of the Federal Register 1 The Electoral College is a...

train_prompts.csv

id	prompt_id	text
3 unique values	3 total values	3 unique values
0000aaaa	2	Aaa bbb ccc.

test_essay.csv

제출 양식

Test set의 id, 해당 데이터가 generated 되었을 확률

```
id,generated
0000aaaa,0.1
1111bbbb,0.9
2222cccc,0.4
...
```

#02 Team Goals

#1 Data augmentation, Training, Inference 과정에서 LLM 활용

#2 매주 캐글 노트북 분석 및 공유

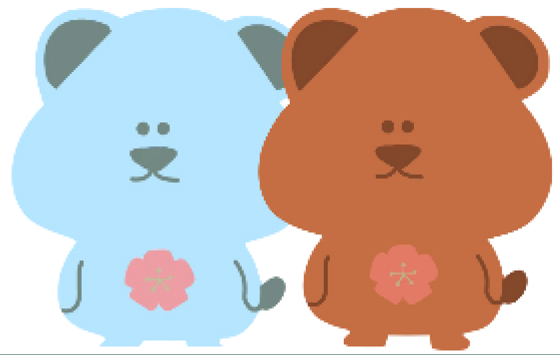
#03 Solutions

#1 LLM finetuning - 정채윤

#2 LLM Ensemble - 황채원

#3 Knowledge Distillation - 배수현

01 Solution 1
- LLM finetuning



EWHA EURON

#01 Mistral-7b

#1 Mistral-7B

MISTRAL AI_

- 상대적으로 적은 수의 파라미터로 LLaMA2 13B, LLaMA1 34B를 많은 수의 벤치마크에서 능가하는 성능을 보인 거대 언어 모델

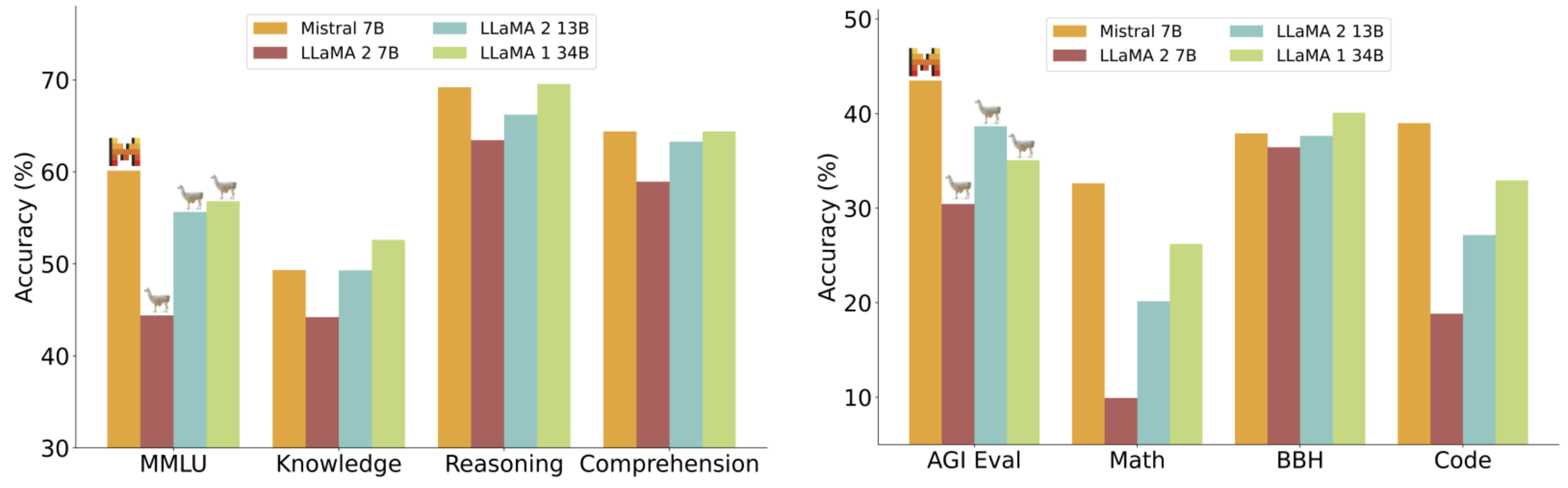


Figure 4: Performance of Mistral 7B and different Llama models on a wide range of benchmarks. All

- Grouped-Query Attention (GQA)

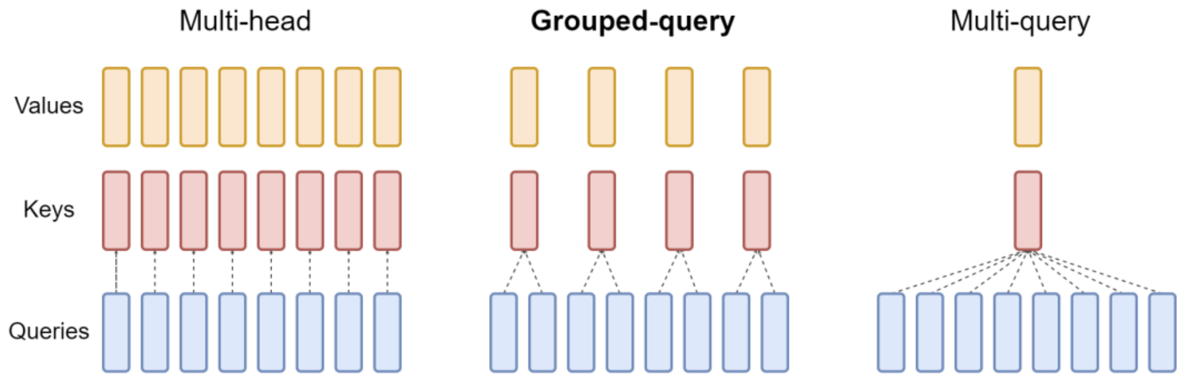
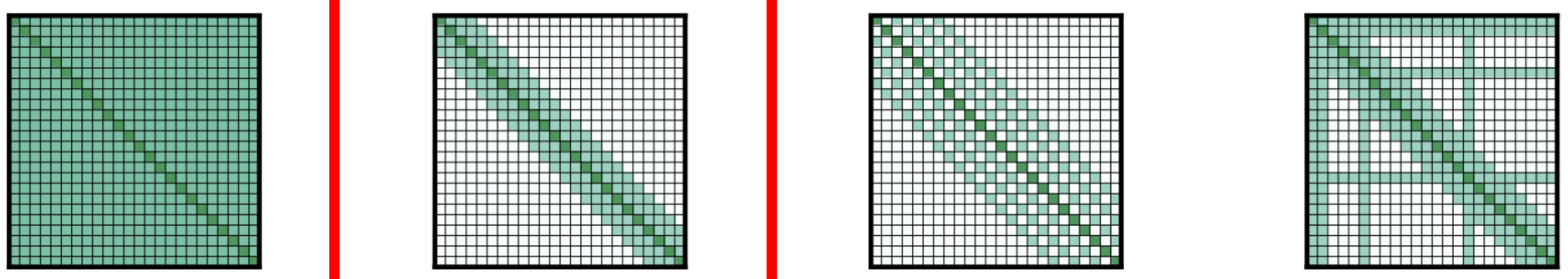


Figure 2: Overview of grouped-query method. Multi-head attention has H query, key, and value heads. Multi-query attention shares single key and value heads across all query heads. Grouped-query attention instead shares single key and value heads for each *group* of query heads, interpolating between multi-head and multi-query attention.

- Sliding-Window Attention (SWA)



(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

Figure 2: Comparing the full self-attention pattern and the configuration of attention patterns in our Longformer.

Jiang et al.(2023), Ainslie(2023), Beltagy et al.(2020)

#02 [GPU train] Mistral-7b | Deberta with optuna

#2 Datasets

- Competition dataset (1,378)
- DAIGT v2 (20,450)
- Slimpajama (1,324,128)

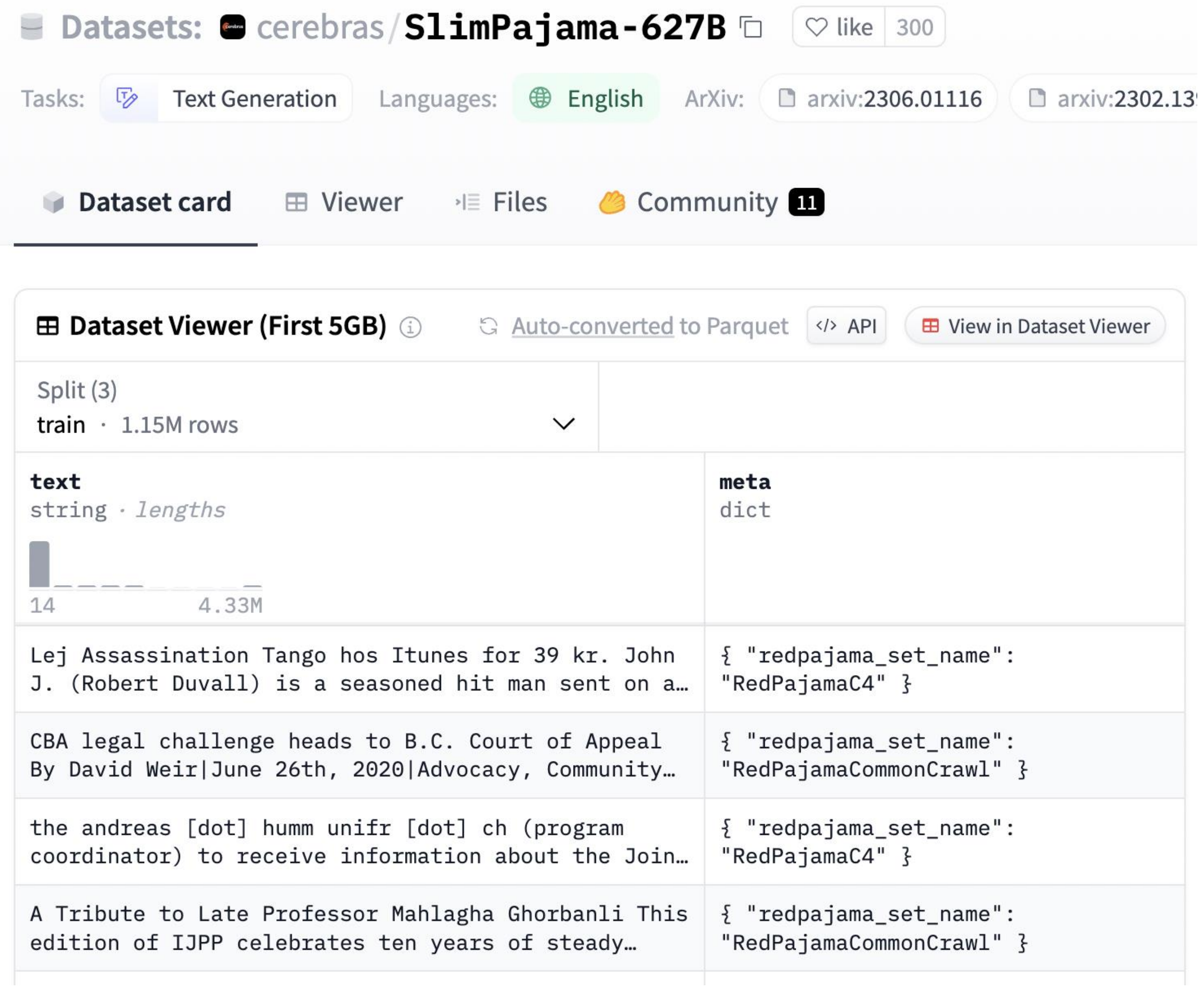
About Dataset

Please use version 2 (there were some issues with v1 that I fixed)!

New release of DAIGT train dataset! Improvement:

- new models: Cohere Command, Google Palm, GPT4 (from Radek!)
- new prompts, including source texts from the original essays!
- mapping of essay text to original prompt from persuade corpus
- filtering by the famous "RDizzl3_seven"

persuade_corpus	25996
chat_gpt_moth	2421
llama2_chat	2421
mistral7binstruct_v2	2421
mistral7binstruct_v1	2421
original_moth	2421
train_essays	1378
llama_70b_v1	1172
falcon_180b_v1	1055
darragh_claude_v7	1000
darragh_claude_v6	1000
radek_500	500
NousResearch/Llama-2-7b-chat-hf	400
mistralai/Mistral-7B-Instruct-v0.1	400
cohere-command	350
palm-text-bison1	349
radekgpt4	200



https://www.kaggle.com/code/minhsienweng/gpu-infer-mistral-7b
https://www.kaggle.com/code/minhsienweng/gpu-train-mistral-7b-deberta-with-optuna

#03 [GPU train] Mistral-7b | Deberta with optuna

#3 Finetuning

- PEFT(Parameter-Efficient FineTuning)
  - LLM 파인튜닝 시 모든 파라미터를 학습시키는 것이 아닌, 일부 적은 수의 파라미터만 학습시켜 계산적 비용과 메모리 비용을 감소시키는 방식
- LoRA(Low-Rank Adaptation)
  - 사전학습된 모델의 가중치를 freeze하고, 학습가능한 rank decomposition matrices를 각 트랜스포머에 추가하여 파라미터 수를 감소시키는 방식

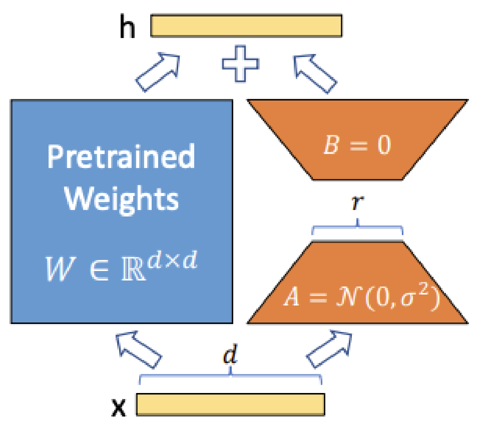


Figure 1: Our reparametrization. We only train A and B.

```python
# LoRa
peft_config = LoraConfig(
    r=self.R,  # Use larger 'r' value increase more parameters during training
    lora_alpha=self.lora_alpha,
    lora_dropout=0.1, # reduce overfitting
    bias='none',
    inference_mode=False,
    task_type=TaskType.SEQ_CLS,
    # Only Use Output and Values Projection
    target_modules=['o_proj', 'v_proj'],
)

# Load the PEFT model
self.model = get_peft_model(base_model, peft_config)
# Display Trainable Parameters to make sure we load the model successfully
self.model.print_trainable_parameters()
print(f"Complete loading pretrained LLM model {time.time() - start:.1f} seconds")
```

```python
def train_model(self):
    # Output folder
    OUTPUT_DIR = "/kaggle/tmp/OUTPUTS"
    # Training arguments
    training_args = TrainingArguments(output_dir=OUTPUT_DIR,
                                      overwrite_output_dir=True,
                                      learning_rate=self.LEARNING_RATE,
                                      per_device_train_batch_size=self.BATCH_SIZE, # A large value runs out of memory
                                      per_device_eval_batch_size=self.BATCH_SIZE, #
                                      num_train_epochs=self.NUM_EPOCHS,
                                      # metric_for_best_model="roc_auc",
                                      push_to_hub=False,
                                      report_to='none',
                                      # optimizer and scheduler
                                      optim='paged_adamw_32bit',
                                      lr_scheduler_type="cosine",
                                      weight_decay=0.01,
                                      max_grad_norm=0.3, # clip global grad norm
                                      gradient_accumulation_steps=16,
                                      # Save the best model only
                                      evaluation_strategy="steps",
                                      save_strategy="steps",
                                      load_best_model_at_end=True,
                                      eval_steps=self.STEPS,
                                      logging_steps=self.STEPS,
                                      seed=SEED,
                                     )
```

https://huggingface.co/docs/peft/index
Hu et al.(2021)

#03 [GPU train] Mistral-7b | Deberta with optuna

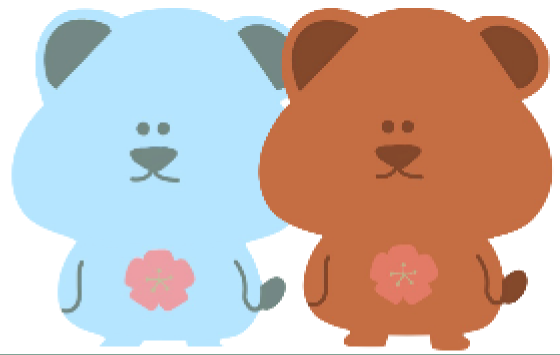
#4 Inference

- Final inference score (private/public): 0.629/0.842

```
# 'model_path' is the path where the original Mistral model is saved
model_path = "/kaggle/input/mistral/pytorch/7b-v0.1-hf/1"  # Mistral"
# Adapter path stores the fine-tuned adapter, generated from the notebook to improve Mistral
model's performance
adpater_path = "/kaggle/input/fine-tuned-mistral-7b/mistral_7b/mistral_7b_GPU"
mistral_inference = MistralModelInference(model_path, adpater_path)

# Example test_texts
test_texts = ["Your test text goes here.", "Another test text."]

# Perform inference
predicted_probs = mistral_inference.inference(test_texts)

# Print the predicted probabilities
print("Predicted Probabilities:", predicted_probs)
```

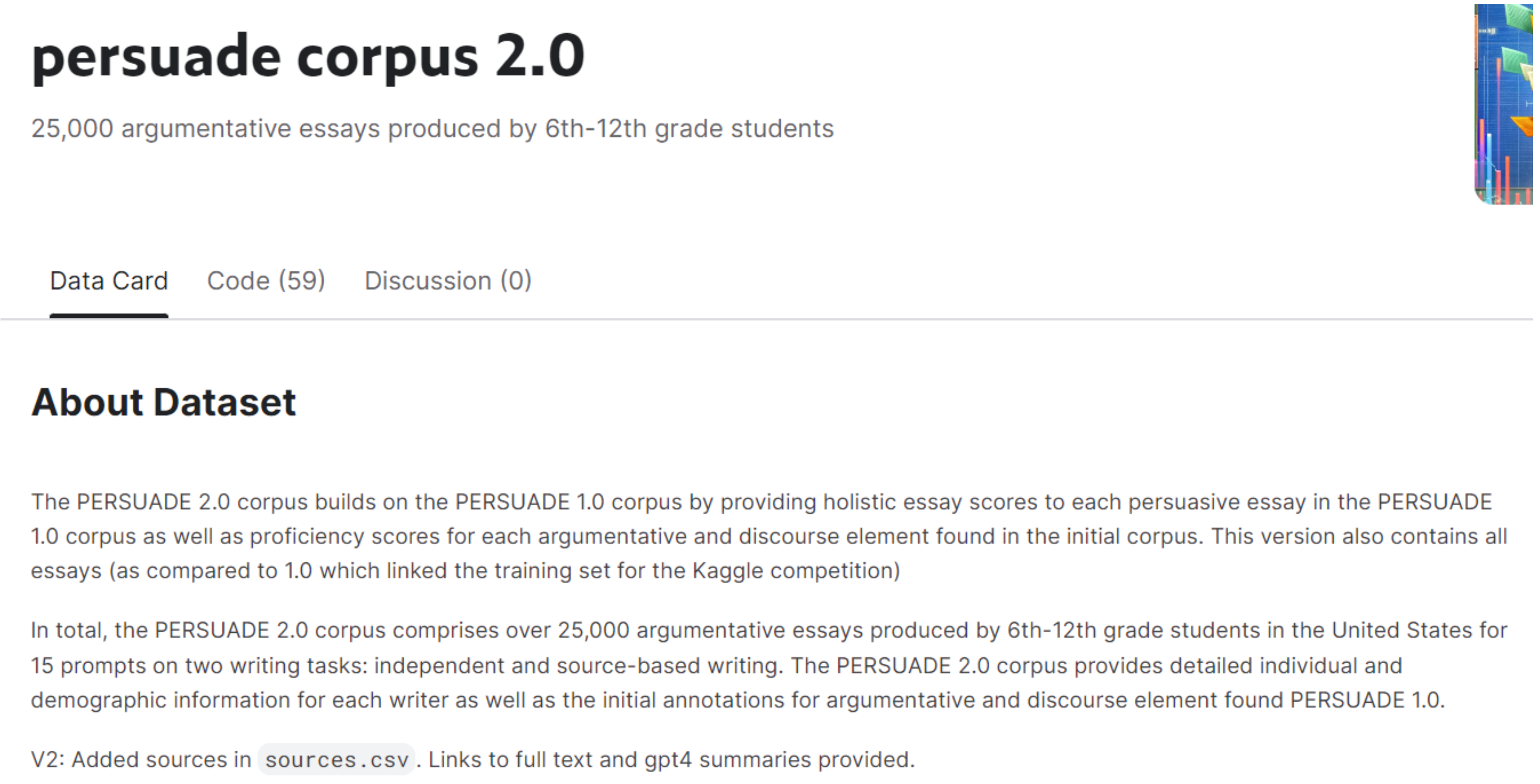
Competition Notebook	Run	Private Score	Public Score
LLM - Detect AI Generated Text	335.9s - GPU P100	0.629009	0.842865

04 Solution 2
- LLM Ensemble



#01 Data Augmentation with finetuned LLM

- Proprietary LLMs (gpt-3.5, gpt-4, claude, cohere, gemini, palm)
- Open source LLMs (llama, falcon, mistral, mixtral)
- Existing LLM generated text datasets
  - Synthetic dataset made by T5
  - DAIGT V2 subset
  - OUTFOX
  - Ghostbuster data
  - gpt-2-output-dataset
- Fine-tuned open-source LLMs (mistral, llama, falcon, deci-lm, t5, pythia, BLOOM, GPT2).



persuade corpus 2.0

25,000 argumentative essays produced by 6th-12th grade students

Data Card Code (59) Discussion (0)

About Dataset

The PERSUADE 2.0 corpus builds on the PERSUADE 1.0 corpus by providing holistic essay scores to each persuasive essay in the PERSUADE 1.0 corpus as well as proficiency scores for each argumentative and discourse element found in the initial corpus. This version also contains all essays (as compared to 1.0 which linked the training set for the Kaggle competition)

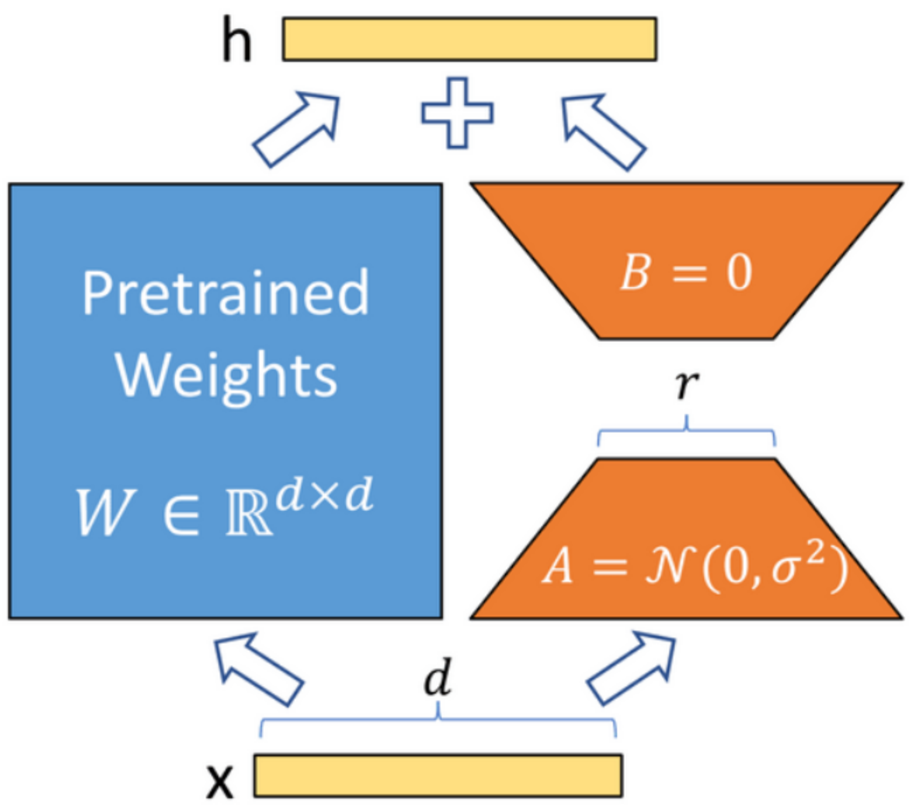
In total, the PERSUADE 2.0 corpus comprises over 25,000 argumentative essays produced by 6th-12th grade students in the United States for 15 prompts on two writing tasks: independent and source-based writing. The PERSUADE 2.0 corpus provides detailed individual and demographic information for each writer as well as the initial annotations for argumentative and discourse element found PERSUADE 1.0.

V2: Added sources in `sources.csv`. Links to full text and gpt4 summaries provided.

#02 Models

Low-Rank Adaptation(LoRA)
: 모델의 모든 파라미터를 업데이트하는 대신 핵심적인 파라미터만을 선택적으로 업데이트하여 효율성을 높임

Quantized Low-Rank Adaptation(QLoRA)
: LoRA에 양자화가 추가된 방식
모델의 가중치나 연산을 더 적은 비트로 표현하여 모델의 크기를 줄이고 계산 효율성을 높인다.



LLM (Q)LoRA fine-tuning: Mistral 7b

We fine-tuned the mistralai/Mistral-7B-v0.1 backbone using (Q)LoRA with config provided below on our carefully curated datamix.

```
peft_config = LoraConfig(
        r=64,
        lora_alpha=16,
        lora_dropout=0.1,
        bias="none",
        task_type=TaskType.SEQ_CLS,
        inference_mode=False,
        target_modules=["q_proj", "k_proj", "v_proj", "o_proj"]
    )
```

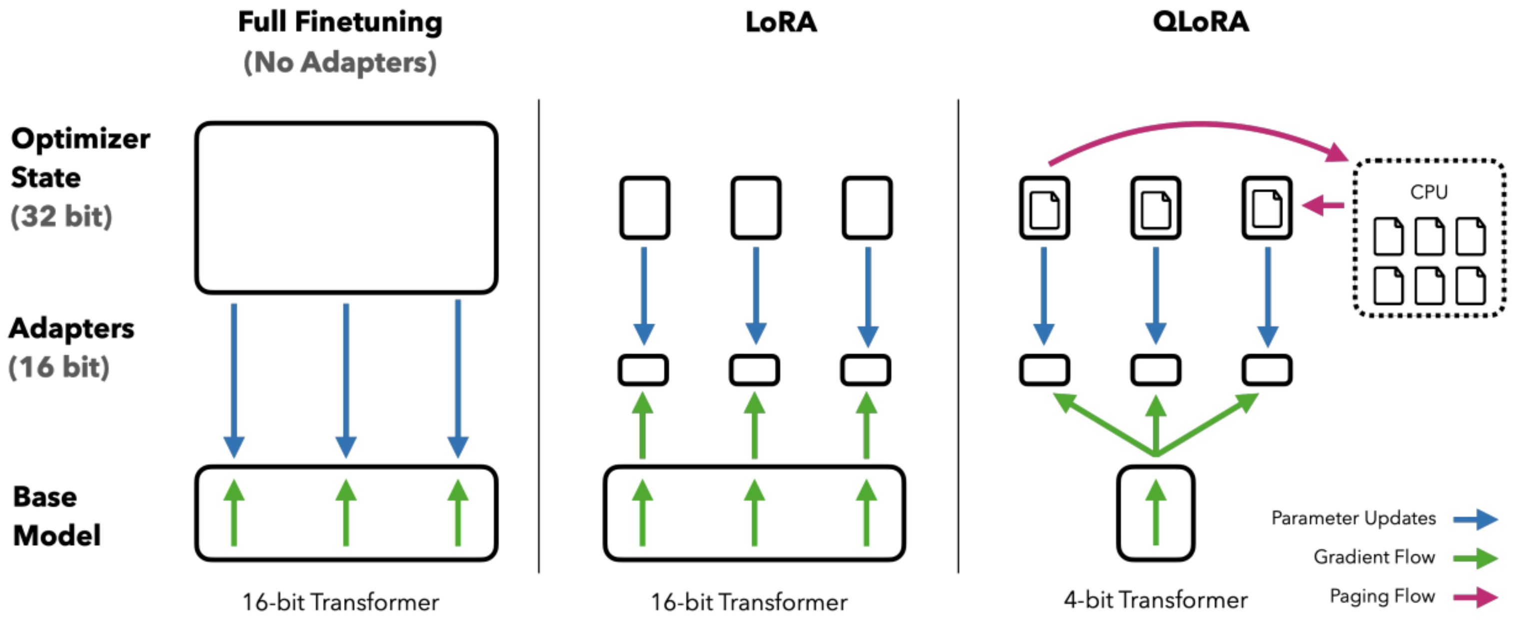
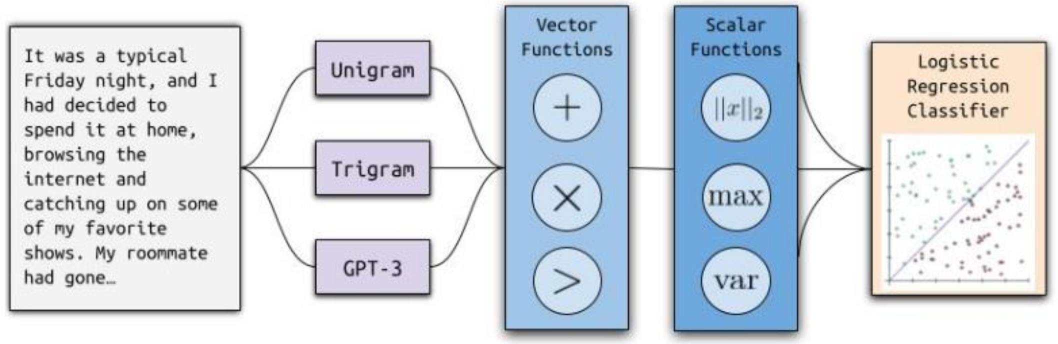


Figure 1: Different finetuning methods and their memory requirements. QLoRA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes.

#02 Models

Ghostbuster: Detecting Text *Ghostwritten* by Large Language Models [paper] [demo] [data]

We introduce Ghostbuster, a state-of-the-art system for detecting AI-generated text. Our method works by passing documents through a series of weaker language models, running a structured search over possible combinations of their features, and then training a classifier on the selected features to predict whether documents are AI-generated.



OpenAI의 davinci와 ada 모델을 사용했지만
여기서는 Llama 7b와 Tiny Llama 1.1B 모델을 사용하여
생성된 토큰의 확률을 추출한다.

- 토큰 확률에 대한 벡터 연산으로 100여개의 피처 생성
- 생성된 피처에 대한 머신러닝 분류 활용

#03 Ensemble

Raw prediction 대신 Ranking을 사용한 앙상블

- 점수가 가장 낮은 텍스트는 순위 1을 받고, 점수가 가장 높은 텍스트는 테스트 세트의 에세이 수(n)만큼의 순위를 받는다.
- 랭킹에 대란 평균을 구한 값이 최종 inference

```python
import pandas as pd
sub_df_m0 = pd.read_parquet("./outputs/m0.parquet")  # mistral
sub_df_m1 = pd.read_parquet("./outputs/m1.parquet")  # mistral

sub_df_m2 = pd.read_parquet("./outputs/m2.parquet")  # deberta-ub

# sub_df_m3 = pd.read_parquet("./outputs/m3.parquet")  # tf-idf

sub_df_m4 = pd.read_parquet("./outputs/m4.parquet")  # pl-deberta


sub_df_m5 = pd.read_parquet("./outputs/m5.parquet")  # ahmet
sub_df_m6 = pd.read_parquet("./outputs/m6.parquet")  # ahmet

sub_df_m7 = pd.read_parquet("./outputs/m7.parquet")  # deberta-rb

sub_df_m8 = pd.read_csv("./outputs/mgb.csv")  # 🐶
```

```python
# # convert to rankings ---
sub_df_m0["generated"] = sub_df_m0["generated"].rank(method='min')
sub_df_m1["generated"] = sub_df_m1["generated"].rank(method='min')
sub_df_m2["generated"] = sub_df_m2["generated"].rank(method='min')
# sub_df_m3["generated"] = sub_df_m3["generated"].rank(method='min')
sub_df_m4["generated"] = sub_df_m4["generated"].rank(method='min')
sub_df_m5["generated"] = sub_df_m5["generated"].rank(method='min')
sub_df_m6["generated"] = sub_df_m6["generated"].rank(method='min')
sub_df_m7["generated"] = sub_df_m7["generated"].rank(method='min')
sub_df_m8["generated"] = sub_df_m8["generated"].rank(method='min')
```

EWHA EURON

05 Solution 3
- Knowledge Distillation

EWHA EURON

#01 Knowledge Distillation

Knowledge distillation

- 큰 모델의 지식을 더 작은 모델로 전달하는 기술

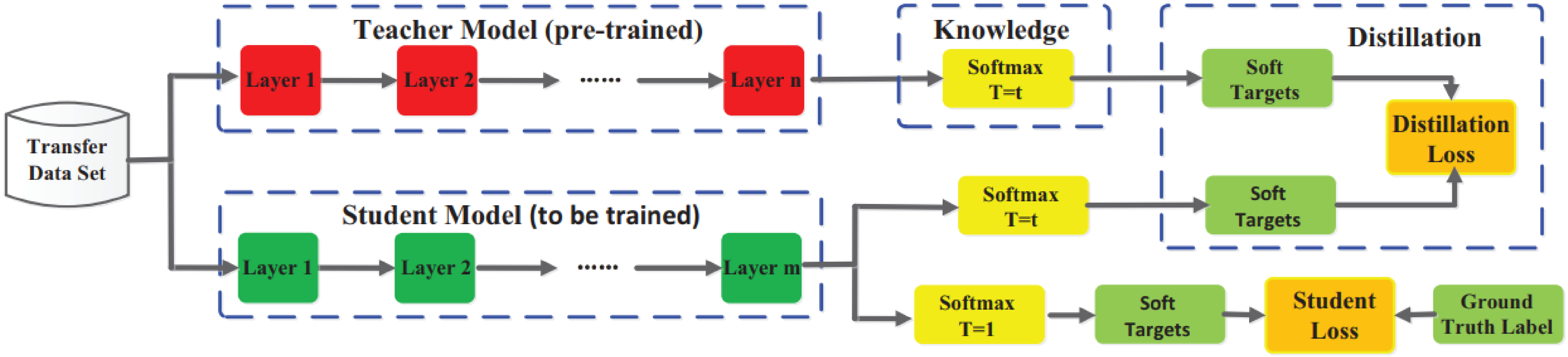


Fig. 5 The specific architecture of the benchmark knowledge distillation (Hinton et al., 2015).

Knowledge Distillation 순서

1. 원본 모델 (teacher model or ensemble) 훈련

2. 작은 모델(student model) 초기화
   - 일반적으로 더 적은 parameter 및 비슷한 layer 구조

3. Student Model은 실제 라벨(ground truth)과 teacher model의 soft target을 사용하여 손실을 계산하고 역전파하여 모델을 훈련

$$\text{Total loss} = \underbrace{(1-\alpha)L_{ce}(\sigma(Z_s), \hat{y})}_{\text{Student Loss}} + 2\alpha T^2 \underbrace{L_{ce}(\sigma(\frac{Z_s}{T}), \sigma(\frac{Z_s}{T}))}_{\text{Distillation Loss}}$$

Data generation 〉 Teacher model finetuning 〉 Teacher model inference 〉 Student model train/infere nce

Overall Pipeline

#02 Data Generation

Original Data

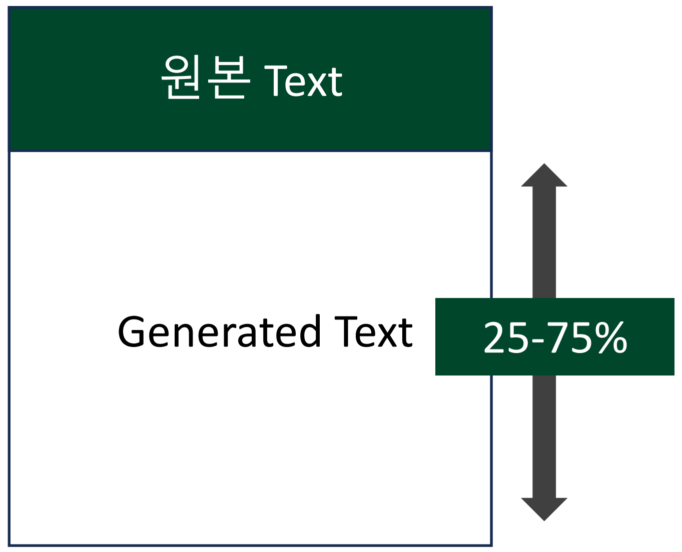
- #1 Pile Completions
- #2 Slim Pajama
- #3 PERSUADE
- #4 Tricky Crawl

Locally hosted LLMs

Model	Document count	Model	Document count
Airoboros-L2-13B-2.1-AWQ	46,358	Mistral-7B-v0.1	61,851
CodeLlama-34B-AWQ	29,885	mpt-7b	12,232
falcon-7b	11,178	OpenHermes-2.5-Mistral-7B	21,221
Llama-2-13B-AWQ	61,531	StableBeluga2-70B-AWQ	21,046
Llama-2-13B-chat-AWQ	43,145	WizardCoder-Python-34B-V1.0-AWQ	41,716
Llama-2-70B-AWQ	31,297	WizardLM-70B-V1.0-AWQ	41,979
Mistral-7B-Instruct	46,825	zephyr-7b-beta	42,093

Generate Data

Ex. Pile Completions



Random crop

Dataset	# Human documents	# Generated documents
PERSUADE essays	25,996	327,268
Uncopyrighted Pile Completions	512,371	512,371
SlimPajama Completions	233,146	233,146
Tricky Crawl	125,192	0

Generated Dataset

#03 Teacher Model Finetuning

Model

#1 DeBERTa

- decoding-enhanced BERT with Disentangled Attention
- enhanced mask decoder
- 같은 정보를 두개의 다른 vector에 저장

#2 Mamba

- 긴 시퀀스 데이터를 효율적으로 모델링하는 새로운 신경망 모델
- transformer 기반 모델들이 긴 시퀀스 처리에서 보여주는 계산 비효율성을 극복하기 위해 설계된 새로운 신경망 구조 제안

Finetuning

```python
# TRAIN AND TEST THE MODEL.
best_auroc = -99999999
train_losses = []
for batch_index, train_batch in enumerate(tqdm(train_data_loader)):
    # SEND DATA TO GPU.
    # Have shape (batch size, token count)
    token_sequences = train_batch.input_ids.cuda()
    attention_masks = train_batch.attention_mask.cuda()
    # Has shape (batch size)
    labels = train_batch.is_artificially_generated.cuda()

    # CLEAR GRADIENTS.
    optimizer.zero_grad()

    # FORWARD PASS.
    with torch.cuda.amp.autocast():
        output = model(token_sequences, attention_masks)
        loss = criterion(output.logits, labels)

    # BACKWARD PASS.
    scaler.scale(loss).backward()

    # UPDATE MODEL PARAMETERS, LR SCHEDULE, ETC.
    scaler.unscale_(optimizer)
    torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)

    scaler.step(optimizer)
    scaler.update()
    lr_schedule.step()

    # RECORD LOSS.
    train_losses.append(loss.detach().cpu())
```

TrainTranformer.py/TrainModel

Optimizer: Adam

Loss function: CrossEntropyLoss

Scaler: GradScaler

batch size: 2 (DeBERTa v3 large)

lrSchedule:torch.optiim.lr_scheduler.

OneCycleLR()

1. deberta: https://huggingface.co/docs/transformers/en/model_doc/deberta
2. Mamba: https://arxiv.org/abs/2312.00752

#04 Teacher model inference

#1 Teacher model (DeBERTa, Mamba) inference

```python
def GeneratePredictions(model, test_dataset):
    # load data
    data_loader = torch.utils.data.DataLoader(
        test_dataset,
        batch_size=4,
        shuffle=False, # 데이터를 섞지 않음
        num_workers=1, # 사용한 cpu 개수
        collate_fn=DataCollatorWithPadding(tokenizer)) # 데이터 패딩 및 배치 처리 함수

    all_predictions = []
    with torch.no_grad(): # disables gradient calculation
        for batch in tqdm(data_loader):
            token_sequences = batch.input_ids.cuda() # 입력 토큰 시퀀스
            attention_masks = batch.attention_mask.cuda() # attention mask

            with torch.cuda.amp.autocast(): # 자동 형변환 사용
                raw_predictions = model(token_sequences, attention_masks).logits

            scaled_predictions = raw_predictions.softmax(dim = 1)[:,1]
            all_predictions.append(scaled_predictions.cpu().numpy())

    all_predictions = np.concatenate(all_predictions)

    return all_predictions

# 'text' 칼럼을 list로 변형하여 SimpleTestDataset의 인자로 넘겨줌
test_dataset = SimpleTestDataset(test_df['text'].tolist(), tokenizer, MAX_SEQUENCE_LENGTH_TOKENS)
# 초기 DeBERTa 모델의 예측을 생성함
initial_deberta_predictions = GeneratePredictions(model, test_dataset)
```

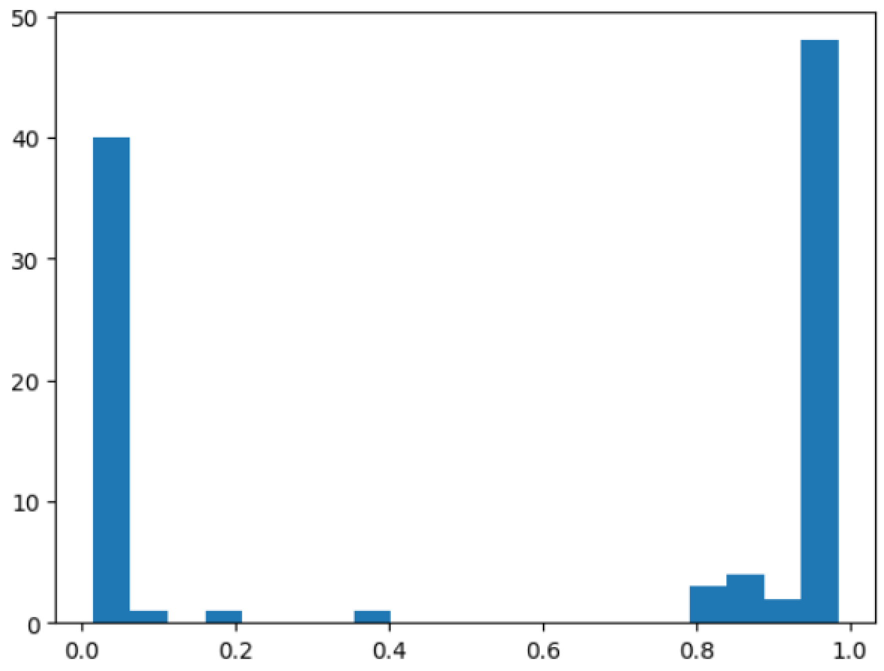
#2 Ensemble Teacher model predictions

```python
initial_deberta_predictions = np.array(initial_deberta_predictions)
initial_mamba_predictions = np.array(initial_mamba_predictions)

initial_predictions = ((0.8*initial_deberta_predictions) + (0.2*initial_mamba_predictions))
```

```python
print(initial_predictions[:5])
```

```
[0.979    0.01611 0.8853   0.0389   0.01778]
```

```python
plt.hist(initial_predictions, bins = 20)
plt.show()
```



#05 Student model train/inference

#3 Generate 1st , 2nd Student model prediction

```
total_transformer_layer_count = len(model.deberta.encoder.layer)
frozen_layer_count = total_transformer_layer_count // 4
print(f'Freezing {frozen_layer_count}/{total_transformer_layer_count} layers!')

for layer in range(frozen_layer_count):
    for name, param in model.deberta.encoder.layer[layer].named_parameters():
        param.requires_grad = False
```

```
Freezing 6/24 layers!
```

- deBERTa v2 모델을 사용하여 encoder layer를 freeze시킴 -> 6/24 개의 layer freeze
- optimizer: AdamW
- Criterion : MSELoss
- scaler: GradScaler()
- train후, inference 진행하여 prediction 생성

#4 Ensemble student model predictions

```
student_1_predictions = np.array(student_1_predictions)
student_2_predictions = np.array(student_2_predictions)

ensemble_predictions = (0.6 * student_1_predictions) + (0.4 * student_2_predictions)
```

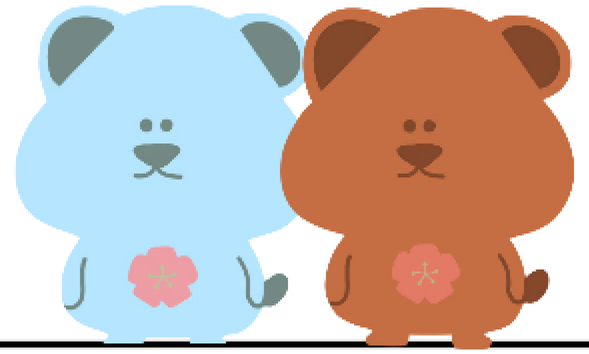
Voting Ensemble 기법

- 모델의 예측 결과를 가중 평균하여 최종 prediction 생성

최종 Score

Competition Notebook	Run	Private Score	Public Score	Best Score
LLM - Detect AI Generated Text	357.6s - GPU T4 ×2	0.969561	0.964687	0.969561 V1

THANK YOU



EWHA EURON