How to use Transformer

Sep 11, 2023

해군 AI 전문인력 양성과정

Jihyoung Jang

Artificial Intelligence Graduate School

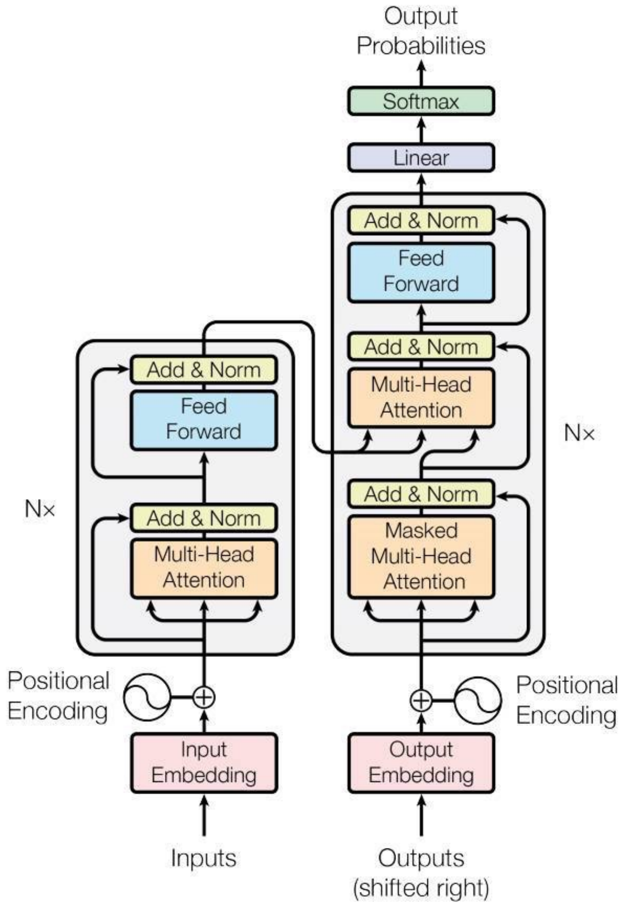
Motivation

Transformer?

* Img src: https://news.tfw2005.com/2023/04/03/transformers-rise-of-the-beasts-new-promotional-poster-479362

Motivation

Transformer?



Vaswani et. al., “Attention Is All You Need,” NeurIPS, 2017

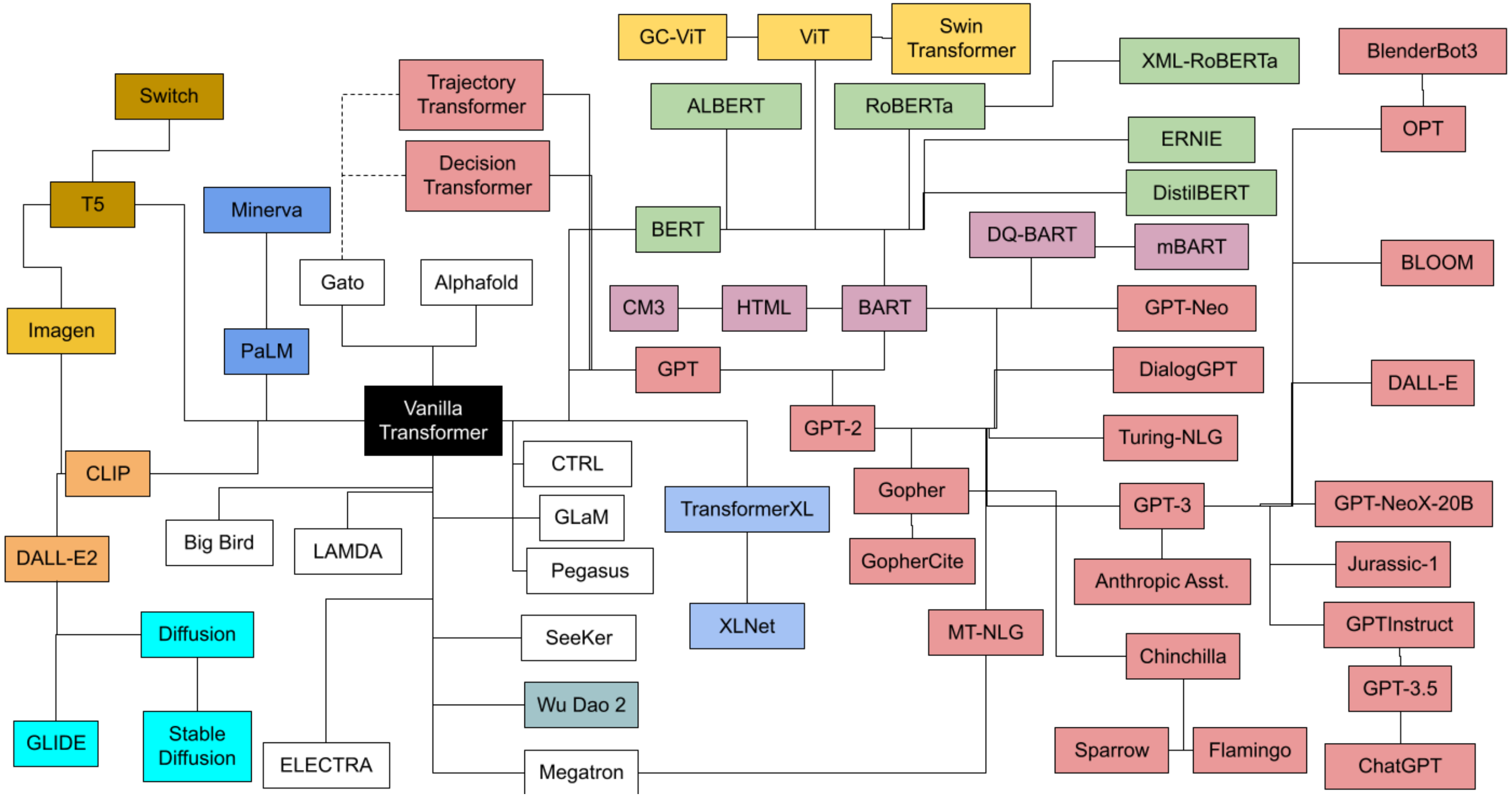
Motivation

Transformer?

* Img src: https://huggingface.co/brand

Transformer

Model Family



* Img src: https://amatriain.net/blog/transformer-models-an-introduction-and-catalog-2d1e9039f376/

Transformer

Hugging Face

- Supported models
  - A total of 203 pre-trained models are officially available
  - Natural Language Processing
  - Computer Vision
  - Audio
  - Multimodal

Transformer

Hugging Face

- Today we are using one of the pre-trained transformer models “BERT”
- Also, today’s task is “Natural Language Inference”
- **Skip the details of the model structure**
- **Focus on how to use the Transformer library**

Task

Natural Language Inference

- Does the first sentence imply the second or not?
  - Premise and Hypothesis

101빌딩 근처에 나름 즐길거리가 많습니다. 101빌딩 근처에서 즐길거리 찾기는 어렵습니다.	Contradiction
101빌딩 근처에 나름 즐길거리가 많습니다. 101빌딩 주변에 젊은이들이 즐길거리가 많습니다.	Neutral
101빌딩 근처에 나름 즐길거리가 많습니다. 101빌딩 부근에서는 여러가지를 즐길수 있습니다.	Entailment

Park et. al., “KLUE: Korean Language Understanding Evaluation,” arXiv:2105.09680, 2021

Code Review

Import Library

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification
from transformers import DataCollatorWithPadding
from transformers import TrainingArguments, Trainer
from datasets import Dataset, load_dataset
import evaluate
```

Code Review

Load Dataset

```
train, valid, test = load_dataset("snli", split=["train[:10%]", "validation[:10%]", "test[:10%]"])
```

```
Dataset({ features: ['premise', 'hypothesis', 'label'], num_rows: 55015 })
```

```
train = train.filter(lambda example: example["label"] in [0, 1, 2])
valid = valid.filter(lambda example: example["label"] in [0, 1, 2])
test = test.filter(lambda example: example["label"] in [0, 1, 2])
```

```
Dataset({ features: ['premise', 'hypothesis', 'label'], num_rows: 54958 })
```

```
tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
```

Code Review

Load Tokenizer

- A person on a horse jumps over a broken down airplane.
- A person is training his horse for a competition.

token																									
ID																									
type																									
attention																									

Code Review

Load Tokenizer

- A person on a horse jumps over a broken down airplane.
- A person is training his horse for a competition.

token	[CLS]	a	person	on	a	horse	jumps	over	a	broken	down	airplane	.	[SEP]	a	person	is	training	his	horse	for	a	competition	.	[SEP]
ID	101	1037	2711	2006	1037	3586	14523	2058	1037	3714	2091	13297	1012	102	1037	2711	2003	2731	2010	3586	2005	1037	2971	1012	102
type																									
attention																									

Code Review

Load Tokenizer

- A person on a horse jumps over a broken down airplane.
- A person is training his horse for a competition.

token	[CLS]	a	person	on	a	horse	jumps	over	a	broken	down	airplane	.	[SEP]	a	person	is	training	his	horse	for	a	competition	.	[SEP]
ID	101	1037	2711	2006	1037	3586	14523	2058	1037	3714	2091	13297	1012	102	1037	2711	2003	2731	2010	3586	2005	1037	2971	1012	102
type	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
attention																									

Code Review

Load Tokenizer

- A person on a horse jumps over a broken down airplane.
- A person is training his horse for a competition.

token	[CLS]	a	person	on	a	horse	jumps	over	a	broken	down	airplane	.	[SEP]	a	person	is	training	his	horse	for	a	competition	.	[SEP]
ID	101	1037	2711	2006	1037	3586	14523	2058	1037	3714	2091	13297	1012	102	1037	2711	2003	2731	2010	3586	2005	1037	2971	1012	102
type	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
attention	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Code Review

Load Tokenizer

```
tokenized_train = train.map(tokenize_function, batched=True, num_proc=8)
tokenized_valid = valid.map(tokenize_function, batched=True, num_proc=8)
tokenized_test = test.map(tokenize_function, batched=True, num_proc=8)
```

```
Dataset({ features: ['premise', 'hypothesis', 'label', 'input_ids', 'token_type_ids’, 'attention_mask'], num_rows: 54958 })
```

Code Review

Data Collator

```
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)

sample_batch = tokenized_train[:16]
sample_batch = {k: v for k, v in sample_batch.items() if k not in ["premise", "hypothesis"]}
[len(x) for x in sample_batch["input_ids"]]
```

```
[25, 28, 24, 15, 13, 13, 26, 26, 25, 47, 37, 39, 18, 16, 17, 27]
```

Code Review

Data Collator

```
sample_batch = data_collator(sample_batch)
{k: v.shape for k, v in sample_batch.items()}

{'input_ids': torch.Size([16, 47]),
'token_type_ids': torch.Size([16, 47]),
'attention_mask': torch.Size([16, 47]),
'labels': torch.Size([16])}
```

Code Review

Load Model

```
model = AutoModelForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=3)

training_args = TrainingArguments(
  output_dir="./snli",
  learning_rate=5e-5,
  per_device_train_batch_size=128,
  per_device_eval_batch_size=128,
  num_train_epochs=4,
  evaluation_strategy="epoch",
  save_strategy="epoch",
  load_best_model_at_end=True,
  fp16=True,
)
```

Code Review

Evaluation Metric

```
import numpy as np

accuracy = evaluate.load("accuracy")

def compute_metrics(eval_pred):
predictions, labels = eval_pred
predictions = np.argmax(predictions, axis=1)
return accuracy.compute(predictions=predictions, references=labels)
```

Code Review

Train

```
trainer = Trainer(
  model=model,
  args=training_args,
  train_dataset=tokenized_train,
  eval_dataset=tokenized_valid,
  tokenizer=tokenizer,
  data_collator=data_collator,
  compute_metrics=compute_metrics,
)

trainer.train()

metric = trainer.evaluate(eval_dataset=tokenized_test)
```

Code Review

Inference

```
inference = tokenizer("Sentence A", "Sentence B", return_tensors="pt").to(device)

outputs = model(**inference)
print(outputs.logits.argmax(dim=-1)[0].tolist())
```

I love the principles of deep learning
I hate the principles of deep learning

What?

I like professor kim
I love professor kim

What?

Practice

GLUE

- General Language Understanding Evaluation benchmark (GLUE), a tool for evaluating and analyzing the performance of models across a diverse range of existing NLU tasks

GLUE Benchmark:
10 datasets on text classification

Single Sentence

- COLA
- SST-2

Pairs of Sentence

- MRPC
- STS-B
- QQP
- MNLI
- QNLI
- RTE
- WNLI

Wang et. al., “GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding,” ICLR, 2019

Practice

Pipelines

```
inference = tokenizer("Sentence A", "Sentence B", return_tensors="pt").to(device)

outputs = model(**inference)
print(outputs.logits.argmax(dim=-1)[0].tolist())



from transformers import pipeline

pipe = pipeline("text-classification", model=model, tokenizer=tokenizer)
pipe("Input Sentence")
```

* Reference: https://huggingface.co/docs/transformers/main_classes/pipelines

UNIST

ULSAN NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY

2007

THANK YOU

FIRST IN CHANGE