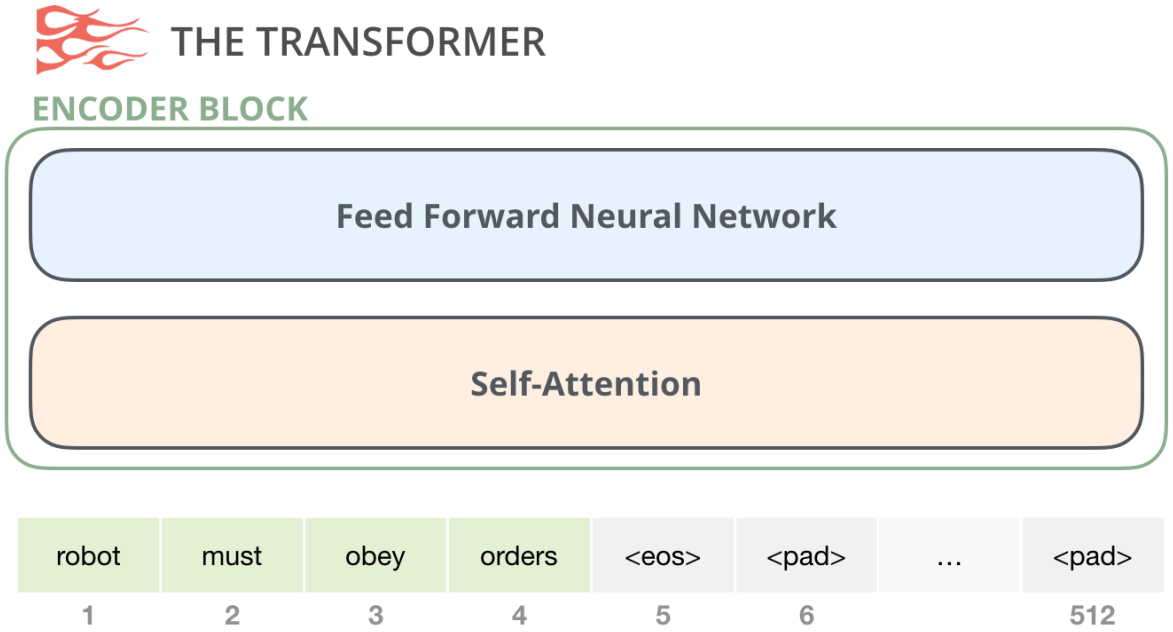
Symbolic Mathematics

KIMIA NOORBAKHSH, MAHDI SHARIFI, MODAR SULAIMAN, SHUGE LEI

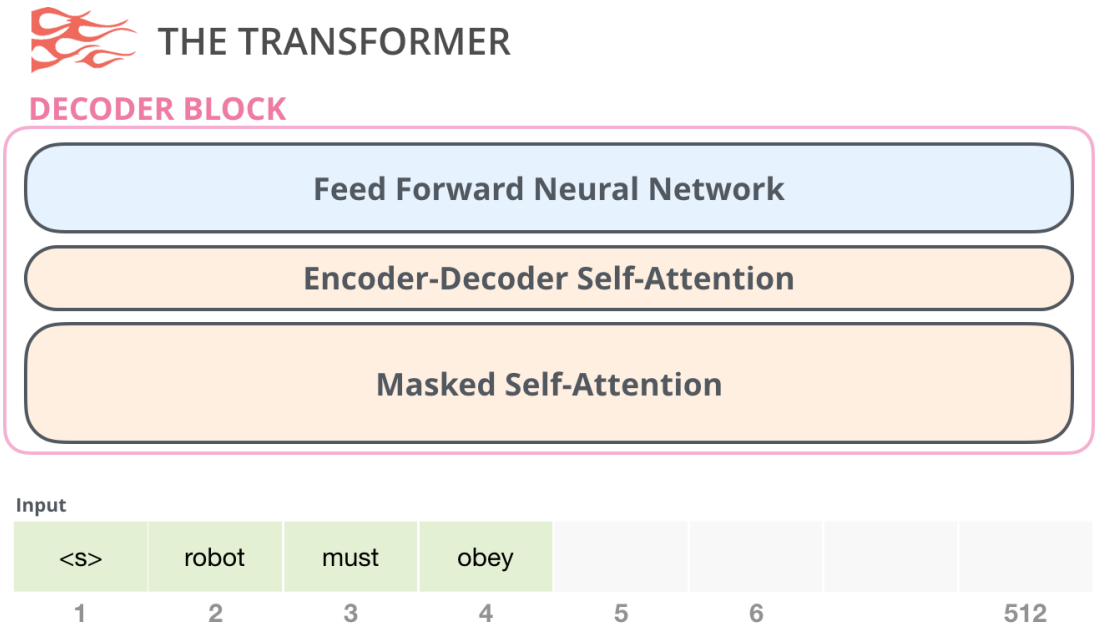
PROF. POOYAN JAMSHIDI, PROF. KALLOL ROY

MAY 31, 2021

Encoder Vs. Decoder

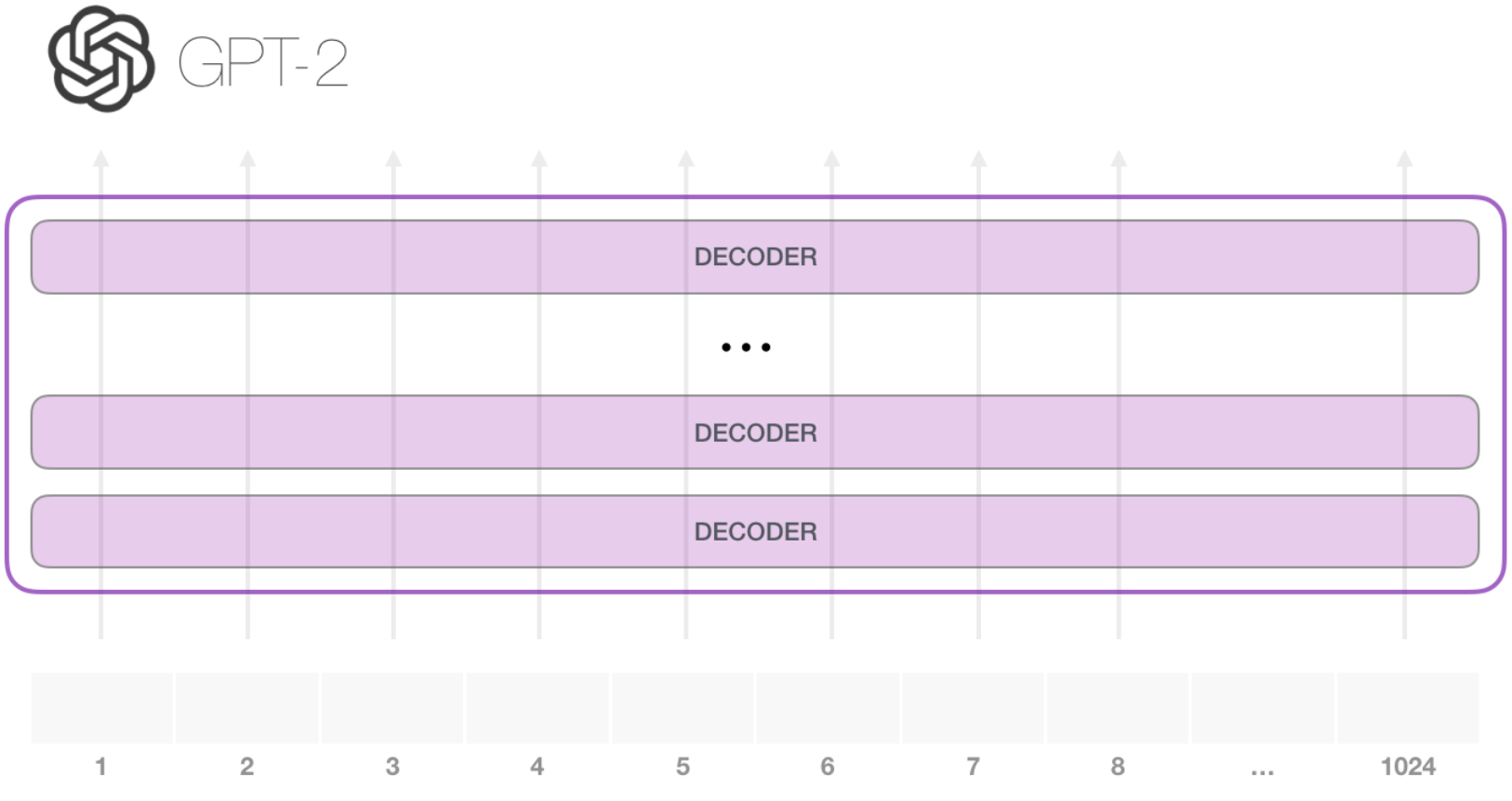


An encoder block from the original transformer paper can take inputs up until a certain max sequence length (e.g. 512 tokens). It's okay if an input sequence is shorter than this limit, we can just pad the rest of the sequence.



The decoder block which has a small architectural variation from the encoder block – a layer to allow it to pay attention to specific segments from the encoder.

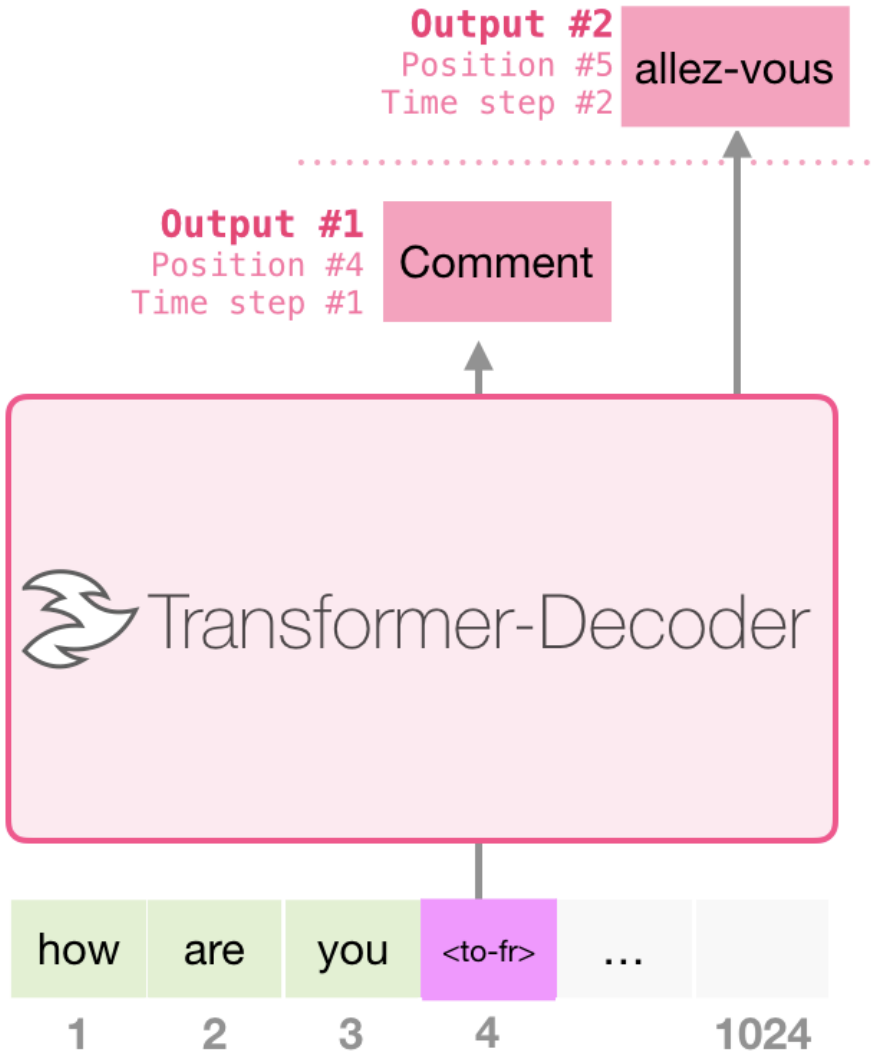
GPT2



Translation

Training Dataset

I	am	a	student	<to-fr>	je	suis	étudiant
let	them	eat	cake	<to-fr>	Qu'ils	mangent	de
good	morning	<to-fr>	Bonjour				



Using the Translation Notebook From Hugging face

Fine-tuning a model on a translation task

In this notebook, we will see how to fine-tune one of the 🤗 Transformers model for a translation task. We will use the WMT dataset, a machine translation dataset composed from a collection of various sources, including news commentaries and parliament proceedings.

Hosted inference API

Translation

My name is Sarah and I live in London | Compute

Computation time on cpu: cached.

Mein Name ist Sarah und ich lebe in London

Their Data


```
show_random_elements(raw_datasets["train"])
```

	translation
0	{'en': 'However, we must not forget that the law has not yet entered into force.', 'ro': 'Cu toate acestea, nu trebuie să uităm că legea respectivă nu a intrat încă în vigoare.'}
1	{'en': 'UniCredit Zagrebacka Banka, based in Bosnia and Herzegovina (BiH) has for the second time won Euromoney magazine's annual award for excellence.', 'ro': 'UniCredit Zagrebacka Banka cu sediul în Bosnia şi Herţegovina (BiH) a câştigat pentru a doua oară premiul anual pentru excelenţă al revistei Euromoney.'}
2	{'en': 'Measuring instruments for cold water meters for non-clean water, alcohol meters, certain weights, tyre pressure gauges and equipment to measure the standard mass of grain or the size of ship tanks have been replaced, in practice, by more modern digital equipment.', 'ro': 'Instrumentele de măsură pentru contoarele de apă rece pentru apa murdară, alcoolmetrele, anumite greutăți, manometrele pentru presiunea din pneuri și echipamentele de măsură pentru masa standard de cereale sau pentru dimensiunea rezervoarelor de nave au fost înlocuite, în practică, de echipamente digitale mai moderne.'}
3	{'en': 'The citizens are our most important allies in achieving our joint objectives.', 'ro': 'Cetăţenii sunt aliaţii noştri cei mai importanţi pentru atingerea obiectivelor noastre comune.'}
4	{'en': 'Nobody can ignore the farmers and our villages because we are not that small.'", 'ro': 'Nimeni nu ne poate ignora, pe noi agricultorii şi satele noastre, pentru că nu suntem atât de mici".'}

The Model to be Fine-Tuned


```
from transformers import AutoModelForSeq2SeqLM, DataCollatorForSeq2Seq, Seq2SeqTrainingArguments, Seq2SeqTrainer

model = AutoModelForSeq2SeqLM.from_pretrained(model_checkpoint)
```

```
batch_size = 16
args = Seq2SeqTrainingArguments(
    "test-translation",
    evaluation_strategy = "epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    weight_decay=0.01,
    save_total_limit=3,
    num_train_epochs=1,
    predict_with_generate=True,
    fp16=True,
)
```

Metric, Bleu Score!


```python
import numpy as np

def postprocess_text(preds, labels):
    preds = [pred.strip() for pred in preds]
    labels = [[label.strip()] for label in labels]

    return preds, labels

def compute_metrics(eval_preds):
    preds, labels = eval_preds
    if isinstance(preds, tuple):
        preds = preds[0]
    decoded_preds = tokenizer.batch_decode(preds, skip_special_tokens=True)

    # Replace -100 in the labels as we can't decode them.
    labels = np.where(labels != -100, labels, tokenizer.pad_token_id)
    decoded_labels = tokenizer.batch_decode(labels, skip_special_tokens=True)

    # Some simple post-processing
    decoded_preds, decoded_labels = postprocess_text(decoded_preds, decoded_labels)

    result = metric.compute(predictions=decoded_preds, references=decoded_labels)
    result = {"bleu": result["score"]}

    prediction_lens = [np.count_nonzero(pred != tokenizer.pad_token_id) for pred in preds]
    result["gen_len"] = np.mean(prediction_lens)
    result = {k: round(v, 4) for k, v in result.items()}
    return result
```

Bleu Score

BLEU Score	Interpretation
< 10	Almost useless
10 - 19	Hard to get the gist
20 - 29	The gist is clear, but has significant grammatical errors
30 - 40	Understandable to good translations
40 - 50	High quality translations
50 - 60	Very high quality, adequate, and fluent translations
> 60	Quality often better than human

Their Results

Then we just need to pass all of this along with our datasets to the `Seq2SeqTrainer`:

```
trainer = Seq2SeqTrainer(
    model,
    args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["validation"],
    data_collator=data_collator,
    tokenizer=tokenizer,
    compute_metrics=compute_metrics
)
```

We can now finetune our model by just calling the `train` method:

```
trainer.train()
```

[38145/38145 1:18:58, Epoch 1/1]

Epoch	Training Loss	Validation Loss	Bleu	Gen Len	Runtime	Samples Per Second
1	0.740100	1.290665	28.059300	34.051500	135.611700	14.741000

```
TrainOutput(global_step=38145, training_loss=0.7717826230230017, metrics={'train_runtime': 4738.3882, 'train_samples_per_second': 8.05,
'total_flos': 3.62019881263104e+16, 'epoch': 1.0, 'init_mem_cpu_alloc_delta': 337319, 'init_mem_gpu_alloc_delta': 300833792, 'init_mem_cp
u_peaked_delta': 18306, 'init_mem_gpu_peaked_delta': 0, 'train_mem_cpu_alloc_delta': 1028051, 'train_mem_gpu_alloc_delta': 899235328, 'tr
ain_mem_cpu_peaked_delta': 267371462, 'train_mem_gpu_peaked_delta': 3136797184})
```

Converting Our data to the same format

	en	ro
0	sub Y' pow x INT+ 2	mul div INT+ 1 INT+ 3 pow x INT+ 3
1	sub Y' add x pow x INT+ 3	add mul div INT+ 1 INT+ 2 pow x INT+ 2 mul div...
2	sub Y' ln x	add mul INT- 1 x mul x ln x
3	sub Y' add x ln x	add mul div INT+ 1 INT+ 2 pow x INT+ 2 add mul...
4	sub Y' pow x div INT+ 5 INT+ 2	mul div INT+ 2 INT+ 7 pow x div INT+ 7 INT+ 2
...	...	...
995	sub Y' mul pow x INT+ 4 mul pow tan INT+ 2 INT...	add mul div INT- 1 INT+ 2 5 mul pow x INT+ 5 p...
996	sub Y' mul pow x INT+ 2 pow add mul INT+ 3 pow...	add mul div INT- 3 INT+ 5 0 ln add INT+ 3 mul ...
997	sub Y' mul INT+ 4 mul pow x INT+ 2 mul cos x s...	mul INT+ 4 mul add mul INT- 2 sin x add mul po...
998	sub Y' add mul INT+ 5 x mul pow x INT- 1 add x...	add x add mul div INT+ 2 INT+ 3 pow x div INT+...
999	sub Y' mul x atanh sqrt sin INT+ 5	mul div INT+ 1 INT+ 2 mul pow x INT+ 2 atanh s...

1000 rows × 2 columns

Took a lot of Pre-Processing! Thanks to Modar.

Results

10 epochs, batch_size = 16, number of samples = 1000 train and 1000 validation.

```
trainer.train()
```

[70/630 03:18 < 27:17, 0.34 it/s, Epoch 1.10/10]

Epoch	Training Loss	Validation Loss	Bleu	Gen Len
1	No log	0.699735	38.175900	34.980000

[630/630 34:44, Epoch 10/10]

Epoch	Training Loss	Validation Loss	Bleu	Gen Len
1	No log	0.699735	38.175900	34.980000
2	No log	0.653457	43.247300	38.805000
3	No log	0.635951	44.151900	39.482000
4	No log	0.640263	40.785600	36.073000
5	No log	0.620414	48.232000	42.115000
6	No log	0.609978	45.758200	38.509000
7	No log	0.598693	48.904800	41.364000
8	0.360000	0.599159	47.634400	40.739000
9	0.360000	0.605166	46.432600	39.321000
10	0.360000	0.604674	47.204200	39.966000

```
TrainOutput(global_step=630, training_loss=0.33667241656591024, metrics={'train_r
```