UNIVERSITY OF South Carolina

UNIVERSITY OF TARTU 1632

Symbolic Mathematics

KIMIA NOORBAKHSH, MAHDI SHARIFI, MODAR SULAIMAN, SHUGE LEI

PROF. POOYAN JAMSHIDI, PROF. KALLOL ROY

MAY 24, 2021

Recall: Generating Dataset

To Generate a random problem or solution:

- Generate a random (unary-binary) tree.
- Randomly select operators as its internal nodes.
- Sample constants or variables as its leaves.

Recall: Generating Dataset

Forward approach (fwd)

- Generate a random function f.
- Compute its integral F with a symbolic framework like SymPy.
- Add (f, F) to dataset.

Functions and their primitives generated with the forward approach (FWD)	
$\cos^{-1}(x)$	$x\cos^{-1}(x) - \sqrt{1-x^2}$
$x\,(2x + \cos(2x))$	$\frac{2x^3}{3} + \frac{x\sin(2x)}{2} + \frac{\cos(2x)}{4}$
$\frac{x\,(x+4)}{x+2}$	$\frac{x^2}{2} + 2x - 4\log(x+2)$
$\frac{\cos(2x)}{\sin(x)}$	$\frac{\log(\cos(x)-1)}{2} - \frac{\log(\cos(x)+1)}{2} + 2\cos(x)$
$3x^2\sinh^{-1}(2x)$	$x^3\sinh^{-1}(2x) - \frac{x^2\sqrt{4x^2+1}}{6} + \frac{\sqrt{4x^2+1}}{12}$

Recall: Generating Dataset

Backward approach (bwd)

- Generate a random function F.
- Compute its derivative f with a symbolic framework like SymPy.
- Add (f, F) to dataset.

Functions and their primitives generated with the backward approach (BWD)	
$\cos(x) + \tan^2(x) + 2$	$x + \sin(x) + \tan(x)$
$\frac{1}{x^2\sqrt{x-1}\sqrt{x+1}}$	$\frac{\sqrt{x-1}\sqrt{x+1}}{x}$
$\left(\frac{2x}{\cos^2(x)} + \tan(x)\right)\tan(x)$	$x\tan^2(x)$
$\frac{x\tan\left(\frac{e^x}{x}\right) + \frac{(x-1)e^x}{\cos^2\left(\frac{e^x}{x}\right)}}{x}$	$x\tan\left(\frac{e^x}{x}\right)$
$1 + \frac{1}{\log(\log(x))} - \frac{1}{\log(x)\log(\log(x))^2}$	$x + \frac{x}{\log(\log(x))}$

Recall: Generating Dataset

Integration By Parts approach (IBP)

- Generate a random functions F and G.
- Compute their derivatives f and g.
- If f * G is in the training set, compute the integral of F * g with

$$\int Fg = FG - \int fG$$

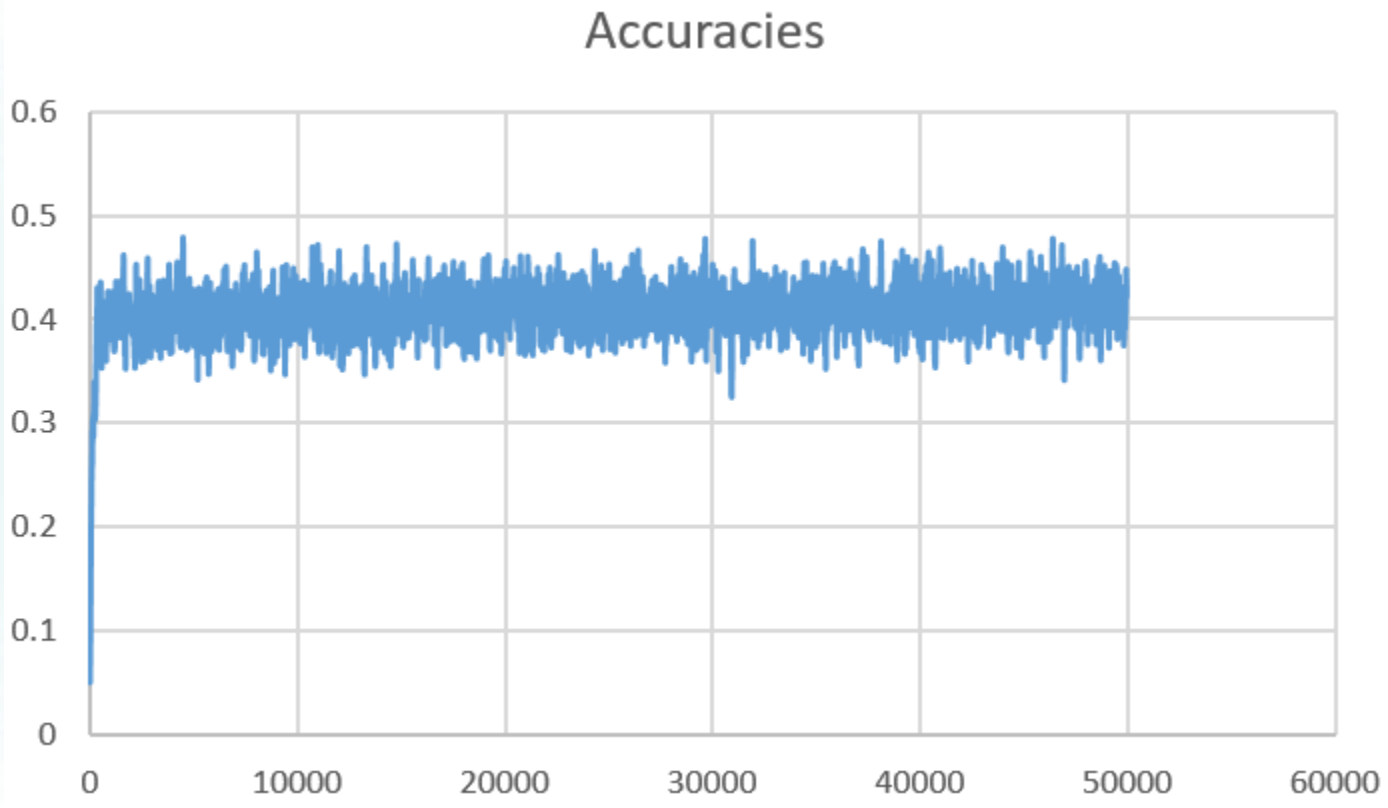
Increased Batch-size


```
gpt2 = GPT2Model.from_pretrained('gpt2')
in_layer = nn.Embedding(len(env.word2id), 768)
out_layer = nn.Linear(768, len(env.word2id))
```

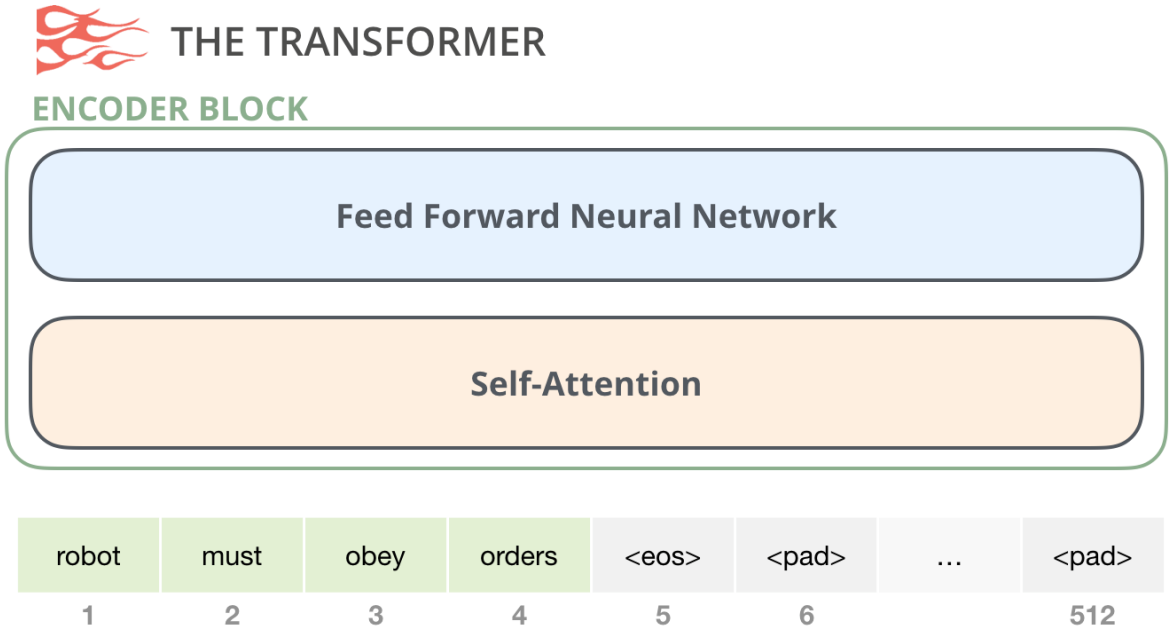
```
embeddings = in_layer(x.reshape(x.shape[1], x.shape[0]))
hidden_state = gpt2(inputs_embeds=embeddings).last_hidden_state[:,:]
logits = out_layer(hidden_state)
logits = logits.reshape(logits.shape[0], logits.shape[2], logits.shape[1])
y = y.reshape(y.shape[1], y.shape[0])
loss = loss_fn(logits, y)
```

Results

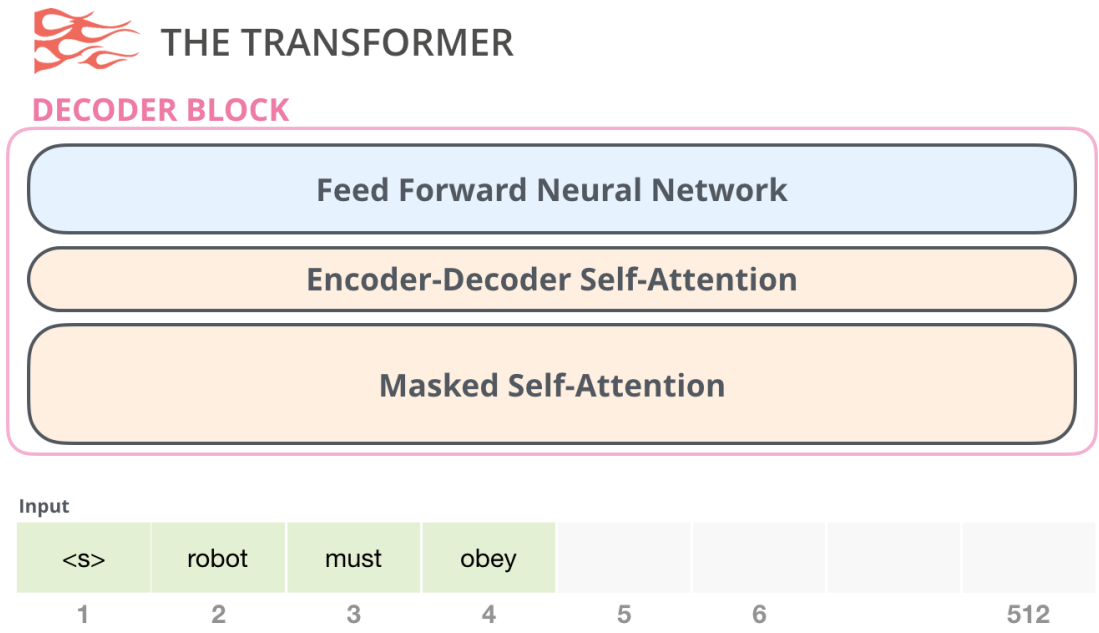
- Dataset: prim_fwd.train
- Here batch_size = 20 and trained on 10000 samples.
- Not going further that 45 %.



Encoder Vs. Decoder

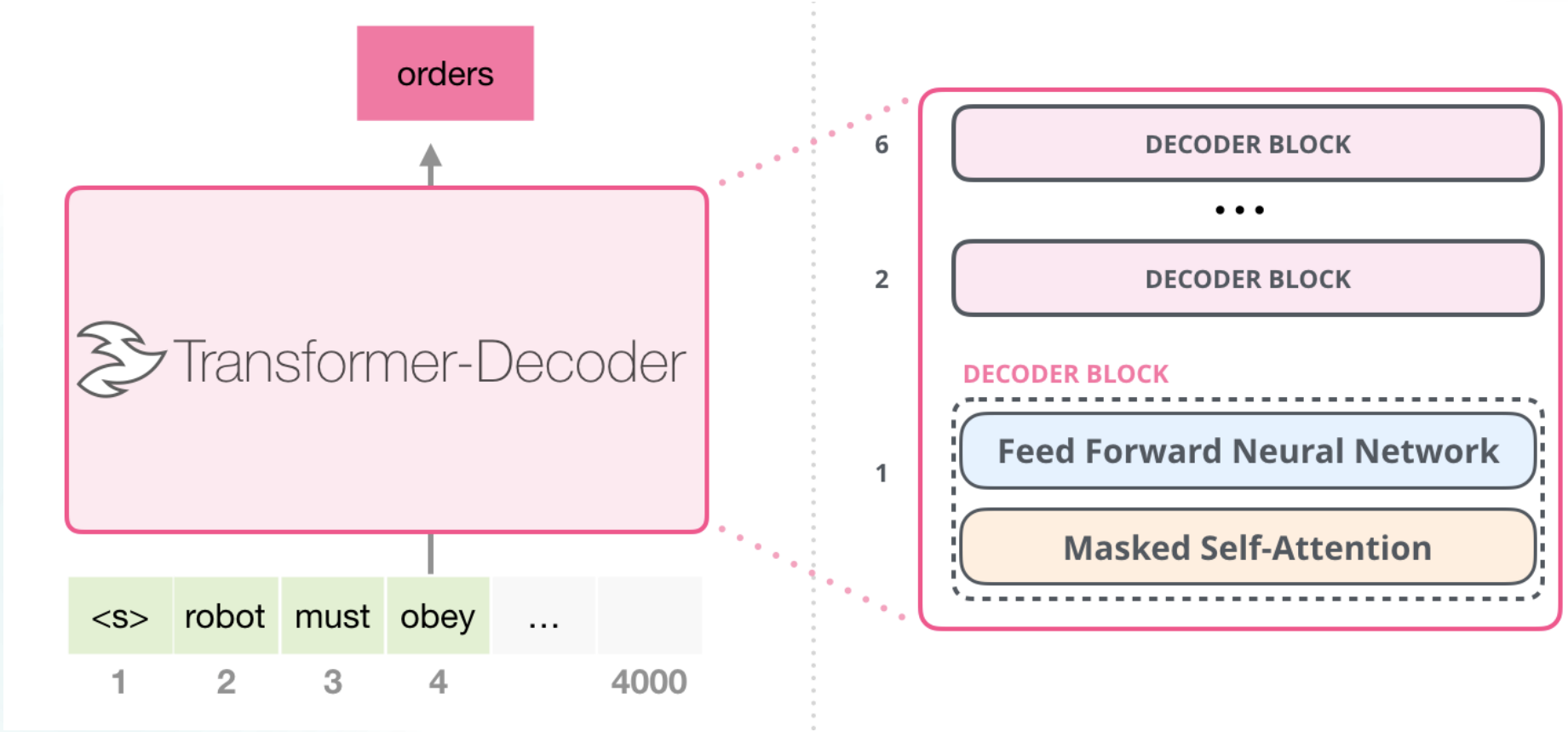


An encoder block from the original transformer paper can take inputs up until a certain max sequence length (e.g. 512 tokens). It's okay if an input sequence is shorter than this limit, we can just pad the rest of the sequence.

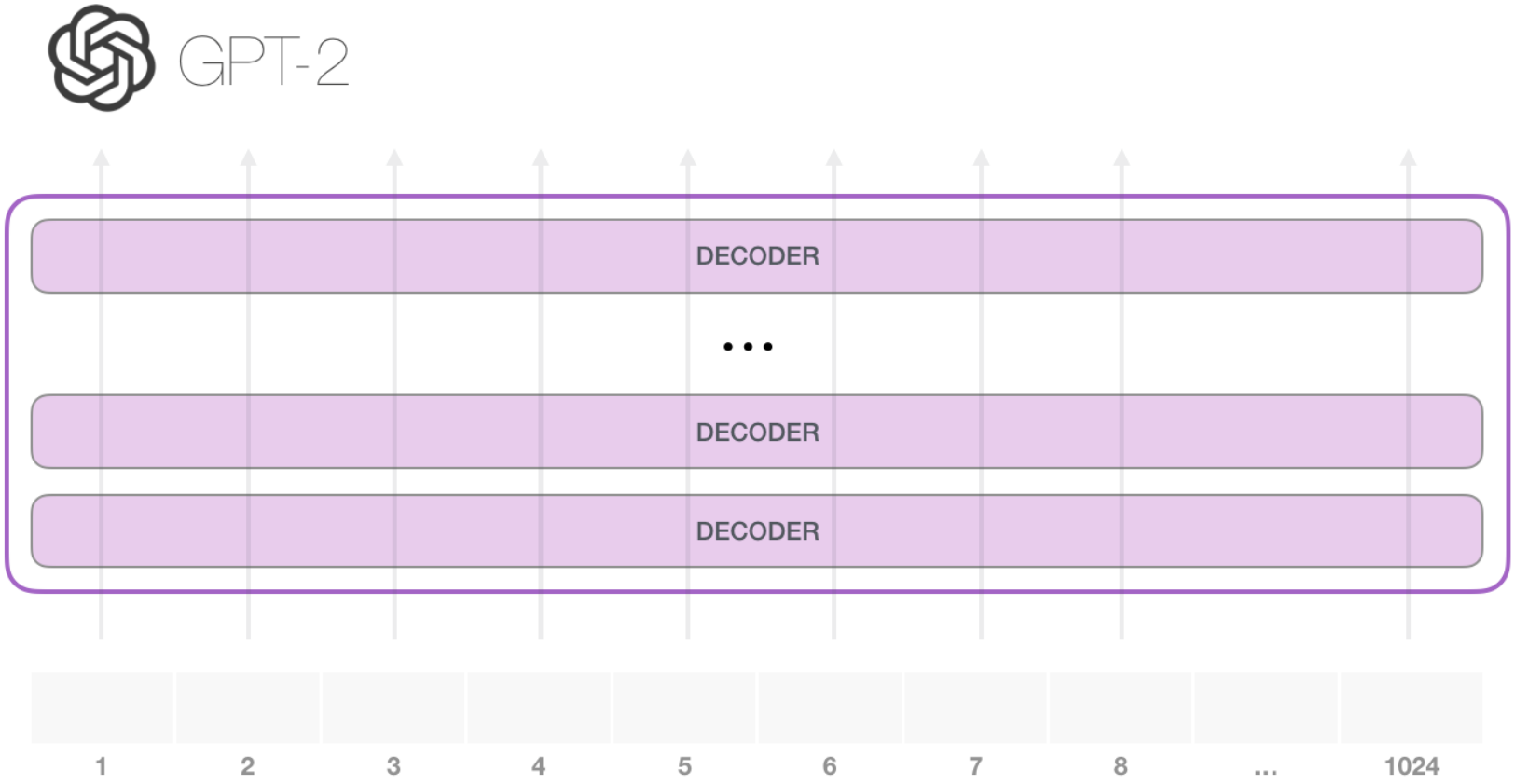


The decoder block which has a small architectural variation from the encoder block – a layer to allow it to pay attention to specific segments from the encoder.

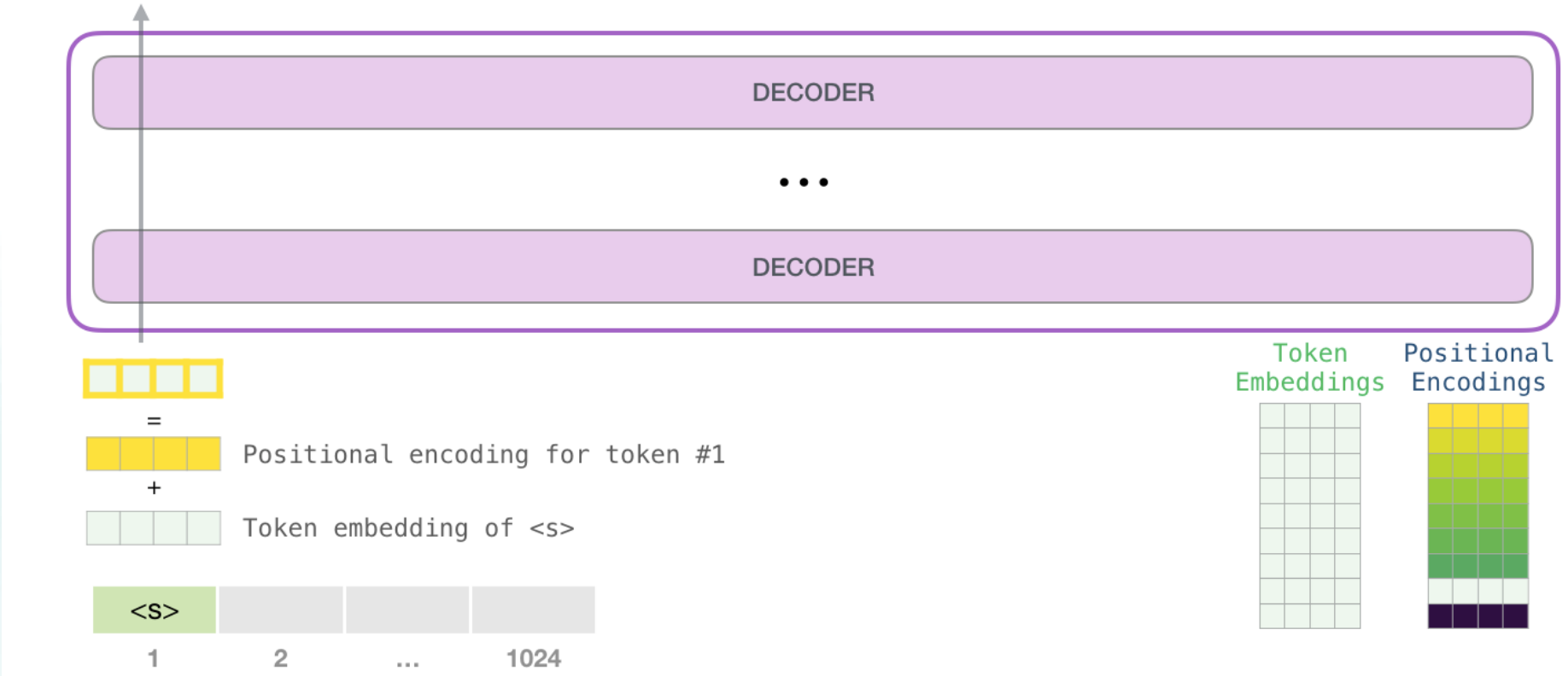
Transformer Decoder



GPT2



GPT2 (Cont.)



Token Embeddings: len(dict) * emb_size
Positional Encodings: 1024 * emb_size

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