Pretrained Transformers As Universal Computation Engines

Kevin Lu, Aditya Grover, Pieter Abbeel, Igor Mordatch

arXiv:2103.05247

The question

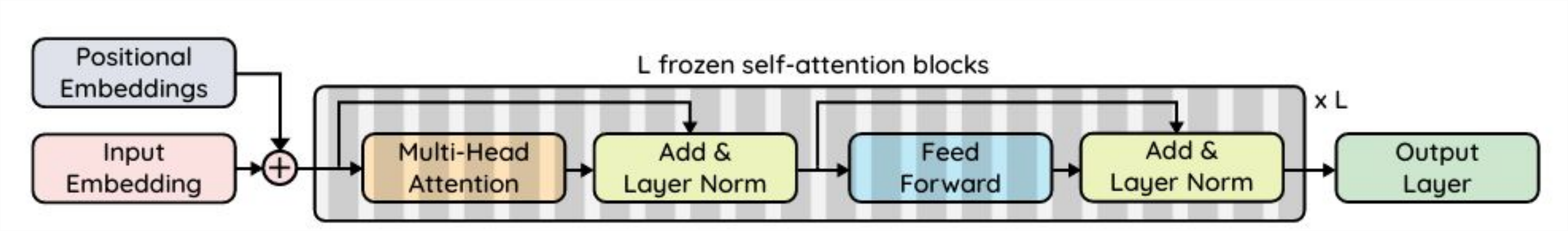
Do pre-trained, state-of-the-art natural language processing models generalize well not just to language-related problems, but also to problems outside of NLP?

Universal computation - architecture

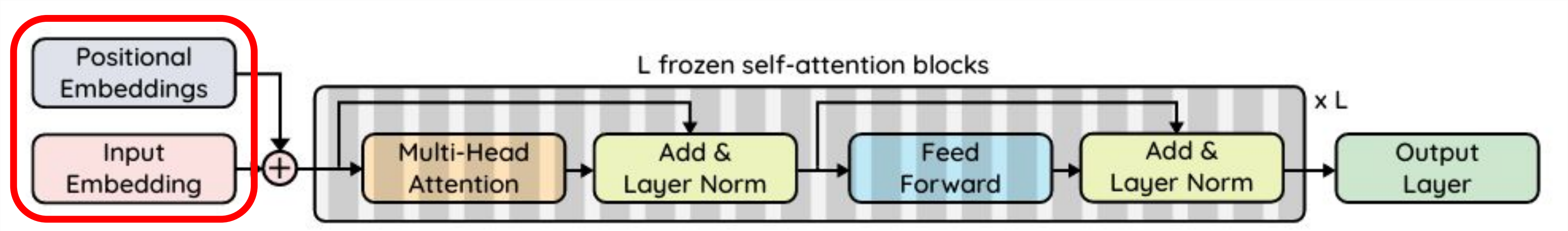
To find the answer, a suitable model is needed first. The authors focused on pre-trained GPT-2 models.

Universal computation - FPT

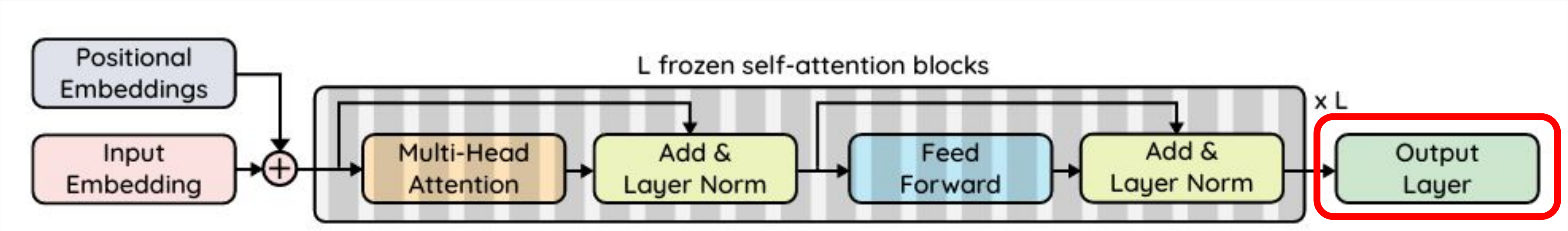
The pretrained GPT-2 model will have most of its parameters frozen (striped on the figure below).



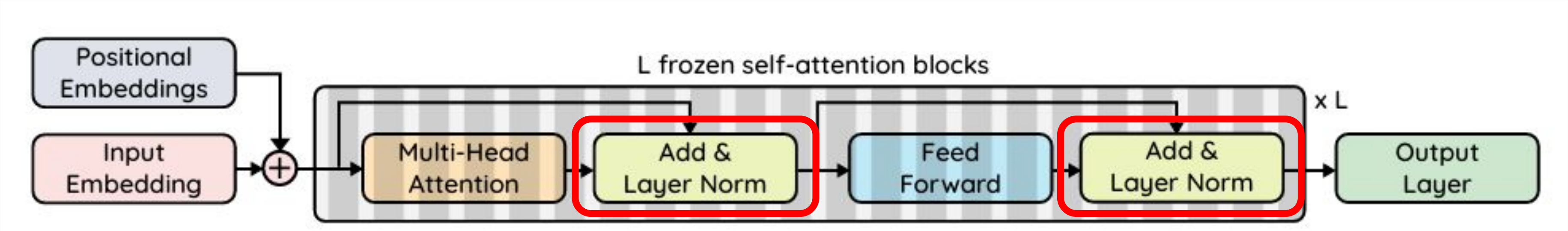
FPT - reinitialized layers



FPT - reinitialized layers



FPT - reinitialized layers



FPT - trainable parameter count

Model Size	n_{layers}	n_{dim}	n_{heads}	# Parameters
Small (Base)	12	768	12	117M
Medium	24	1024	16	345M
Large	36	1280	20	774M

Model Size	# Layers	Total Params	Trained Params
Small (Base)	12	117M	106K
Medium	24	345M	190K
Large	36	774M	300K

Evaluation tasks

The next step is to choose suitable problems to evaluate the model’s capabilities.

The benchmark consists of 7 tasks.

Evaluation tasks - 1. Bit memory

Input: concatenation of 5 bitstrings of length 1000 split into 20 tokens of dimension 50 each; a 6th bitstring which is created by taking a random one of the previous 5 and randomly masking 50% of the bits

Output: recreation of the masked bitstring

Evaluation tasks - 2. Bit XOR

Input: concatenation of 2 bitstrings of length 5

Output: a bitstring of length 5 which is the XOR of the 2 input bitstrings

Evaluation tasks - 3. ListOps

Input: sequence of list operations resulting in a single digit

```
[ MAX 4 3 [ MIN 2 3 ] 1 0 ]
```

Output: the result of the input operations

```
4
```

Evaluation tasks - 4. MNIST

Input: sequence of 4x4 patches over a 32x32 black-and-white image of a handwritten digit

Output: the classification of the input digit

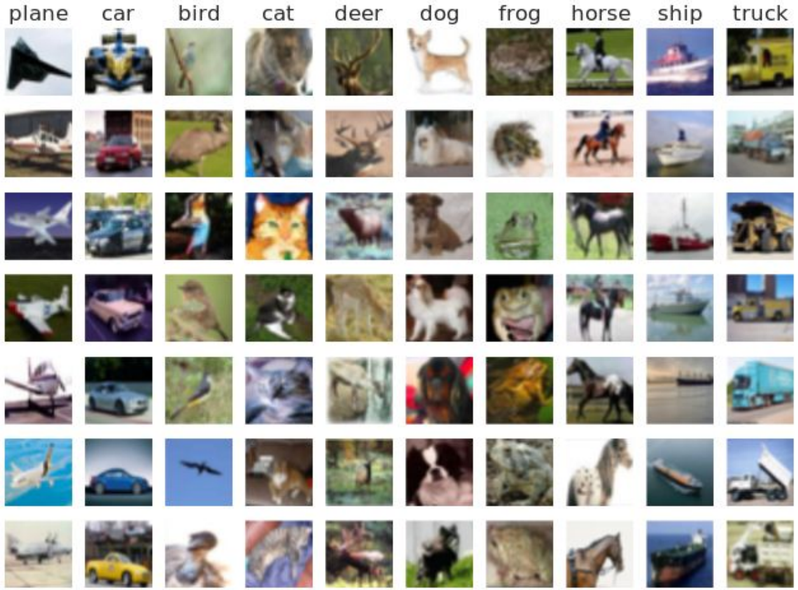
Evaluation tasks - 4. MNIST

Evaluation tasks - 5. CIFAR-10

Input: sequence of 4x4 patches over a 32x32 colored images of 10 different classes

Output: the classification of the input image

Evaluation tasks - 5. CIFAR-10



Evaluation tasks - 6. CIFAR-10 LRA

Input: sequence of 1024 greyscaled pixels from CIFAR-10 images

Output: the classification of the input image as in CIFAR-10

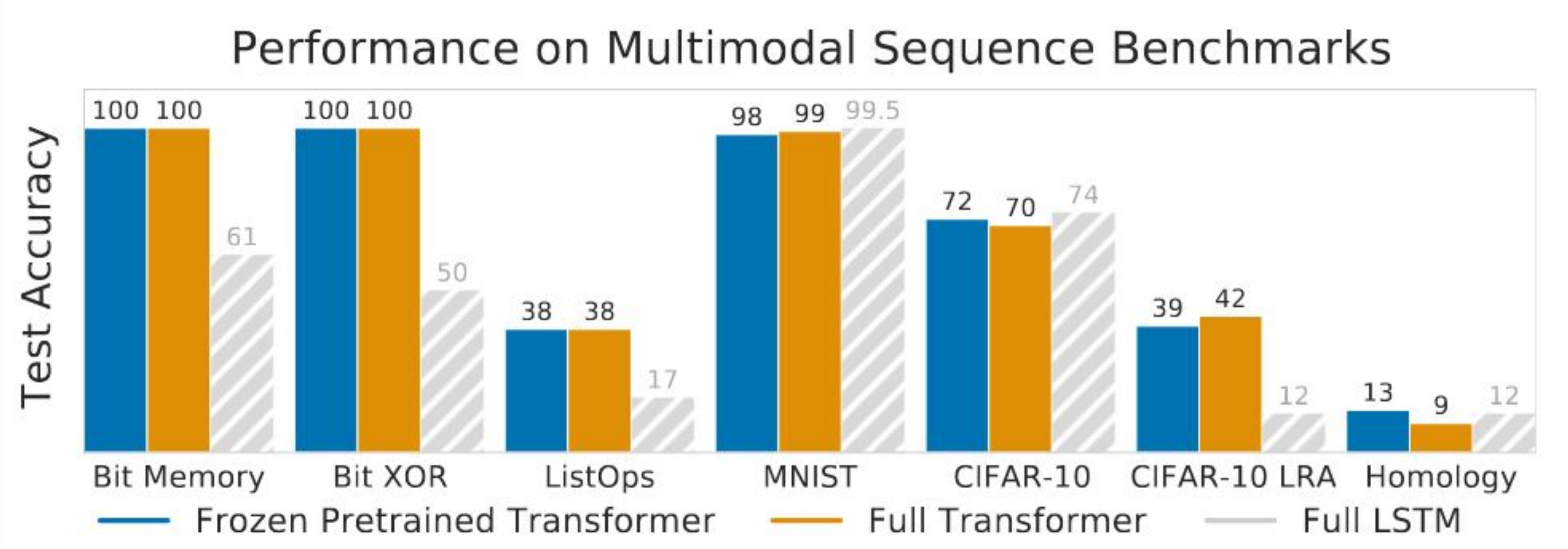
Evaluation tasks - 7. Homology

Input: sequence of up to 1024 aminoacids making up a protein

Output: the classification into one of 1195 classes based on folding

Experiments and results

FPT performance



FPT performance - table

Model	Bit Memory	XOR	ListOps	MNIST	CIFAR-10	C10 LRA	Homology
FPT	100%	100%	38.4%	98.0%	72.1%	38.6%	12.7%
Full	100%	100%	38%	99.1%	70.3%	42%	9%
LSTM	60.9%	50.1%	17.1%	99.5%	73.6%	11.7%	12%

Importance of language pre-training

Model	Bit Memory	XOR	ListOps	MNIST	C10	C10 LRA	Homology
FPT	100%	100%	38.4%	98.0%	68.2%	38.6%	12.7%
Random	75.8%	100%	34.3%	91.7%	61.7%	36.1%	9.3%
Bit	100%	100%	35.4%	97.8%	62.6%	36.7%	7.8%
ViT	100%	100%	37.4%	97.8%	72.5%	43.0%	7.5%

Linear classifier on raw MNIST: 92% accuracy.

Importance of architecture: random weights

Model	Bit Memory	XOR	ListOps	MNIST	CIFAR-10	C10 LRA	Homology
Trans.	75.8%	100%	34.3%	91.7%	61.7%	36.1%	9.3%
LSTM	50.9%	50.0%	16.8%	70.9%	34.4%	10.4%	6.6%

Training duration

Model	Memory	XOR	ListOps	MNIST	C10	C10 LRA	Homology
FPT	1×10^4	5×10^2	2×10^3	5×10^3	4×10^5	3×10^5	1×10^5
Random	4×10^4	2×10^4	6×10^3	2×10^4	4×10^5	6×10^5	1×10^5
Speedup	$4\times$	$40\times$	$3\times$	$4\times$	$1\times$	$2\times$	$1\times$

Over- and underfitting

CIFAR-10 LRA

Model	# Layers	Test Accuracy	Train Accuracy
FPT (GPT-2)	12	38.6%	38.5%
Vanilla Transformer	3	42%	70%
Linformer	3	39%	97%

Scaling with size

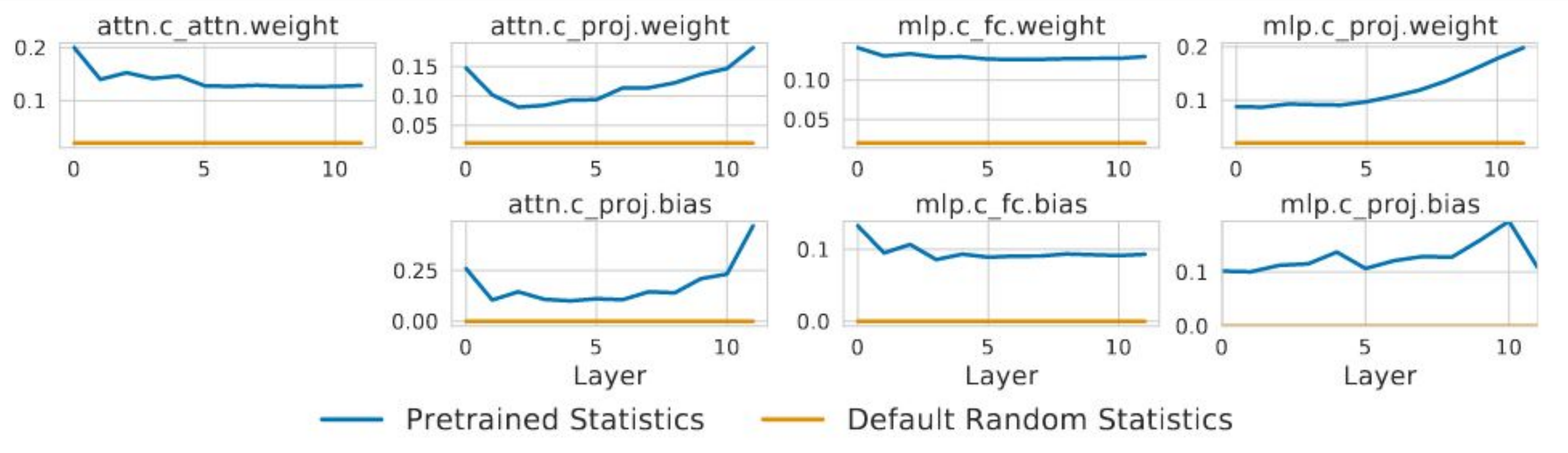
CIFAR-10

Model Size	# Layers	Total Params	Trained Params	FPT	Random
Small (Base)	12	117M	106K	68.2%	61.7%
Medium	24	345M	190K	69.8%	64.0%
Large	36	774M	300K	72.1%	65.7%

Weight initialization

Initialization	Memory	XOR	ListOps	MNIST	C10	C10 LRA	Homology
Pretrained	100%	100%	38.4%	98.0%	68.2%	38.6%	12.7%
Statistics Only	100%	100%	37.4%	97.2%	56.5%	33.1%	11.0%
Default	75.8%	100%	34.3%	91.7%	61.7%	36.1%	9.3%

Weight STDs per layer in FPT



Freezing more parameters

Task	Speedup	Output Only	FPT	Full Transformer
ListOps	$500 - 2000\times$	32.8%	38.4%	38%
CIFAR-10 LRA	$500 - 2000\times$	24.7%	38.6%	42%

Inputs to the last layer are cached for fast training.

Freezing more parameters

Task	output only	output + input	output + positions	output + layernorm
Bit Memory	76%	**98%**	93%	94%
Bit XOR	56%	72%	84%	**98%**
ListOps	15%	17%	35%	**36%**
MNIST	23%	85%	93%	**96%**
CIFAR-10	25%	53%	38%	**54%**
CIFAR-10 LRA	17%	22%	30%	**39%**
Homology	2%	8%	8%	**9%**

Successive finetuning

Task	output only	+ layernorm	+ input	+ positions
Bit Memory	76%	94%	100%	100%
Bit XOR	56%	98%	98%	100%
ListOps	15%	36%	36%	38%
MNIST	23%	96%	98%	98%
CIFAR-10	25%	54%	60%	68%
CIFAR-10 LRA	17%	39%	39%	39%
Homology	2%	9%	10%	13%

Freezing more parameters - frozen random

Task	output only	output + input	output + positions	output + layernorm
Bit Memory	75%	75%	75%	75%
Bit XOR	50%	51%	59%	**100%**
ListOps	17%	17%	18%	**35%**
MNIST	25%	28%	34%	**83%**
CIFAR-10	20%	24%	21%	**46%**
CIFAR-10 LRA	11%	16%	12%	**34%**
Homology	2%	2%	6%	**9%**

Successive finetuning - frozen random

Task	output only	+ layernorm	+ input	+ positions
Bit Memory	75%	75%	75%	76%
Bit XOR	50%	100%	100%	100%
ListOps	17%	35%	36%	37%
MNIST	25%	83%	92%	92%
CIFAR-10	20%	46%	56%	62%
CIFAR-10 LRA	11%	34%	36%	36%
Homology	2%	9%	9%	9%

Freezing fewer parameters

Model	Memory	XOR	ListOps	MNIST	C10	C10 LRA	Homology
FPT	100%	100%	38.4%	98.0%	68.2%	38.6%	12.7%
+ Feedforward	100%	100%	36.0%	98.3%	76.6%	38.2%	13.1%
+ Attention	100%	100%	36.8%	89.0%†	47.7%†	23.0%	10.9%
+ Both	100%	100%	35.8%	93.1%†	32.9%	21.0%	10.5%

Back to our question!

The question

Do pre-trained, state-of-the-art natural language processing models generalize well not just to language-related problems, but also to problems outside of NLP?

The question

Do pre-trained, state-of-the-art natural language processing models generalize well not just to language-related problems, but also to problems outside of NLP?

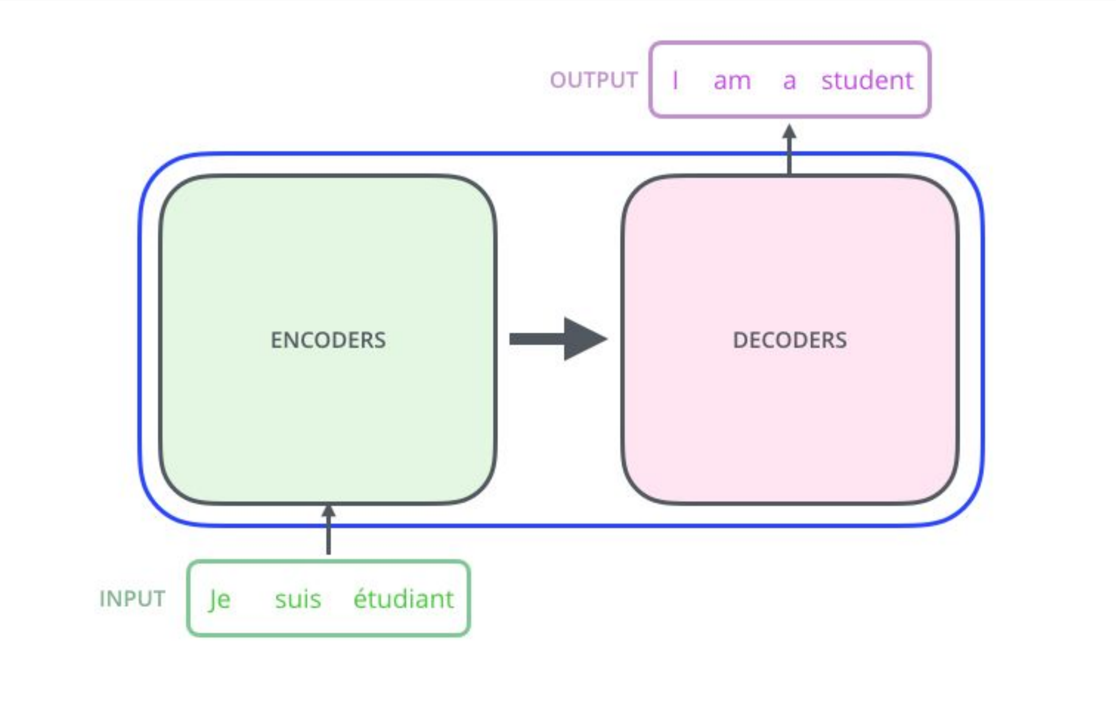
Apparently! But why?

Factor I: the architecture

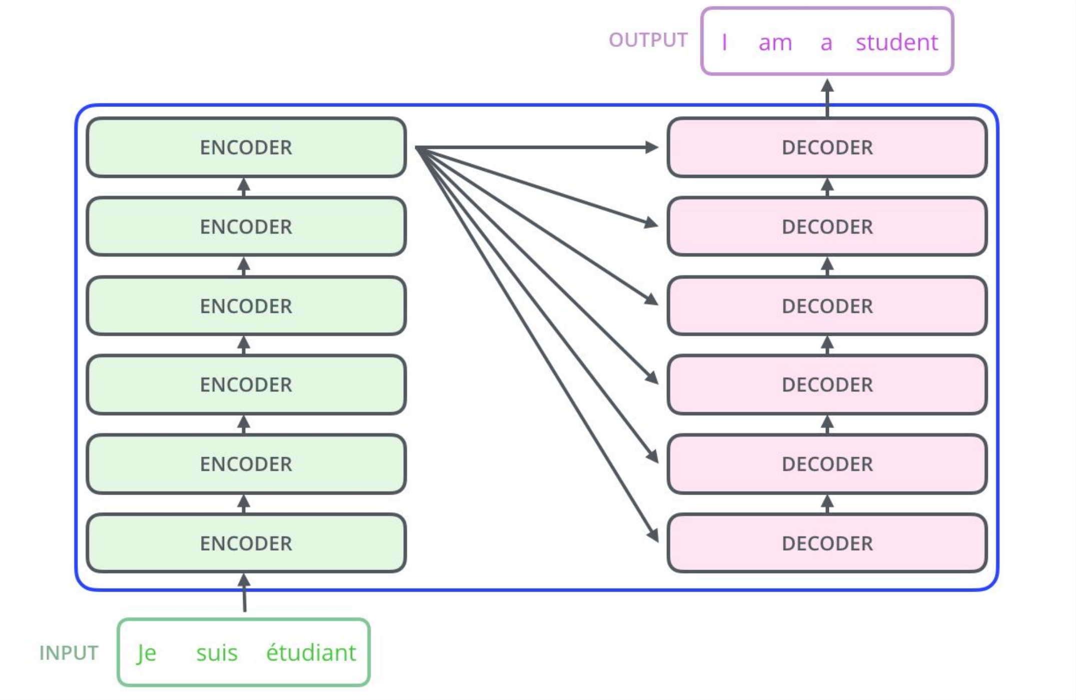
The architecture

To what degree does the Transformer based architecture, especially the self-attention mechanism, enable FPT's performance?

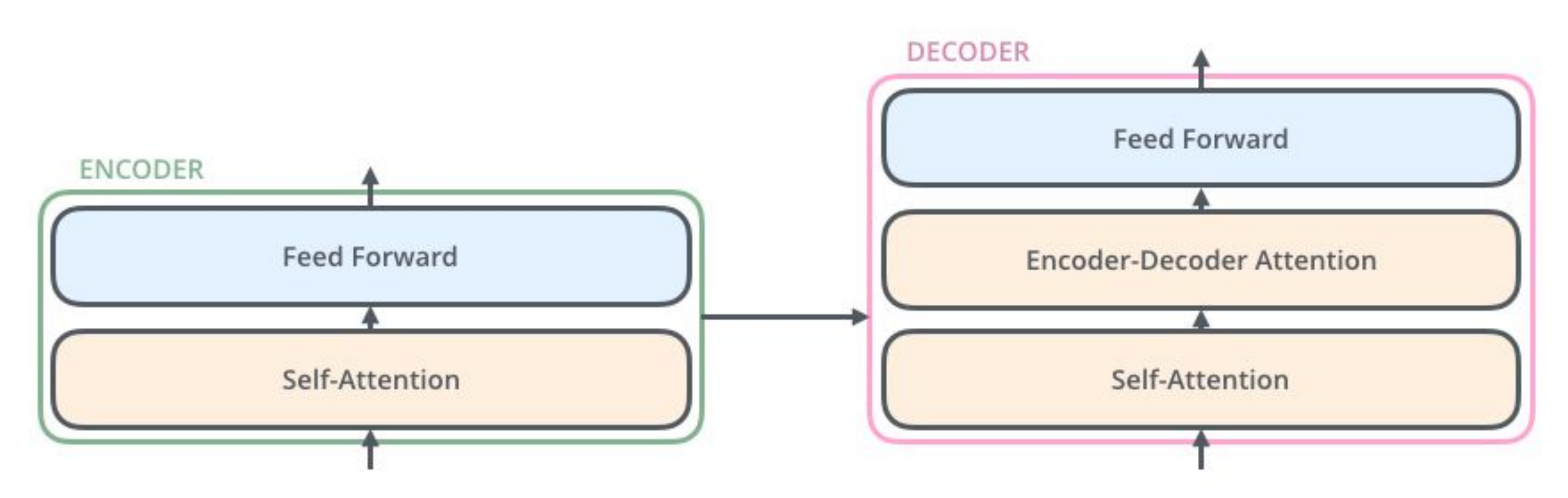
Transformer



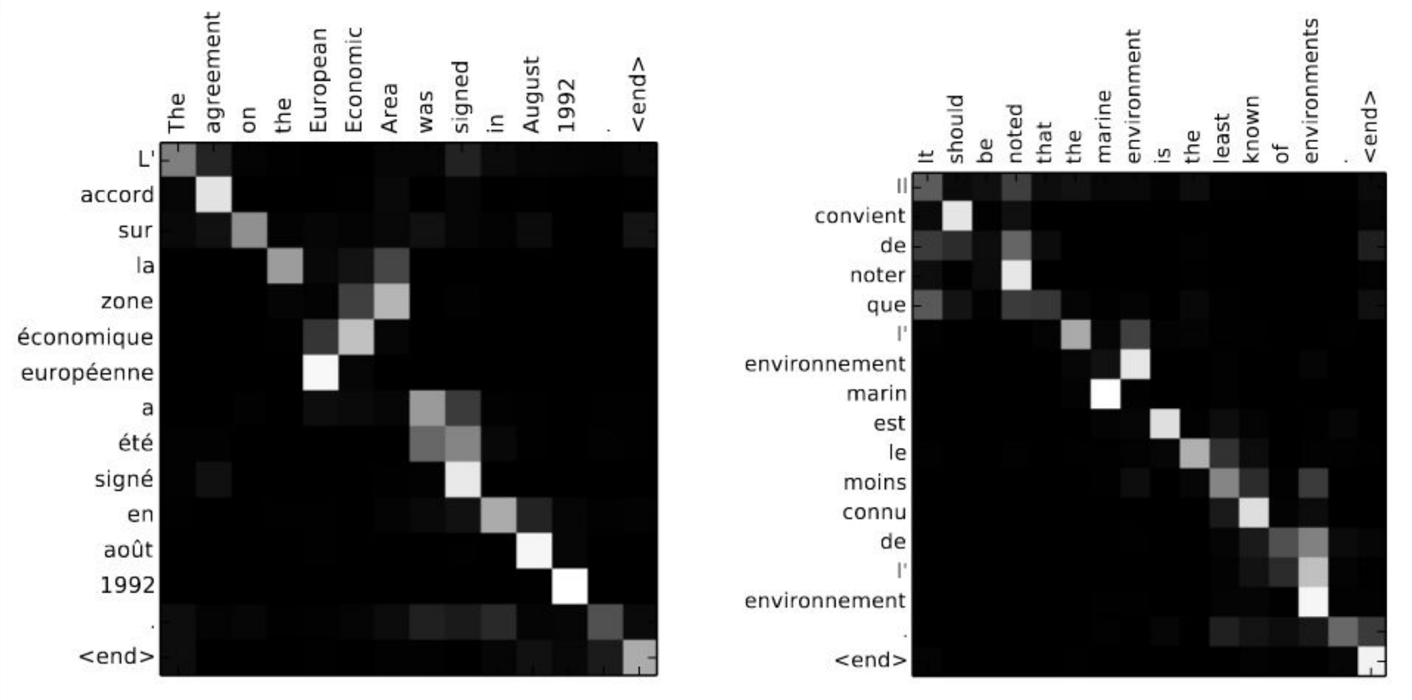
Transformer



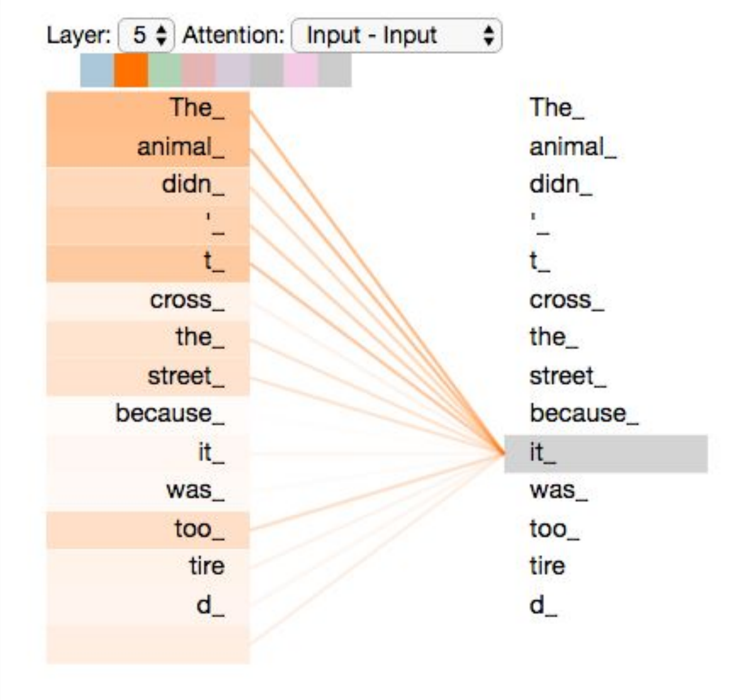
Transformer



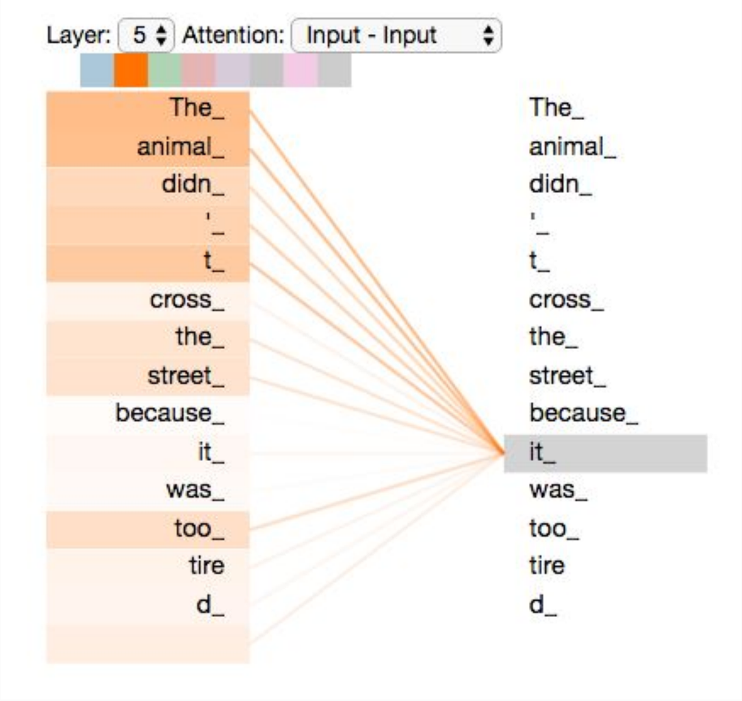
Attention mechanism

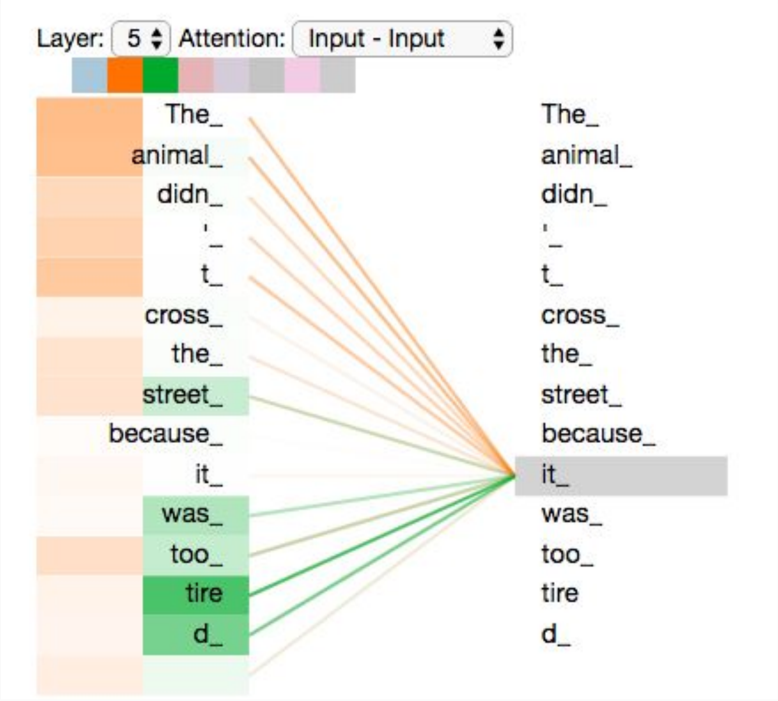


Transformer - self-attention mechanism



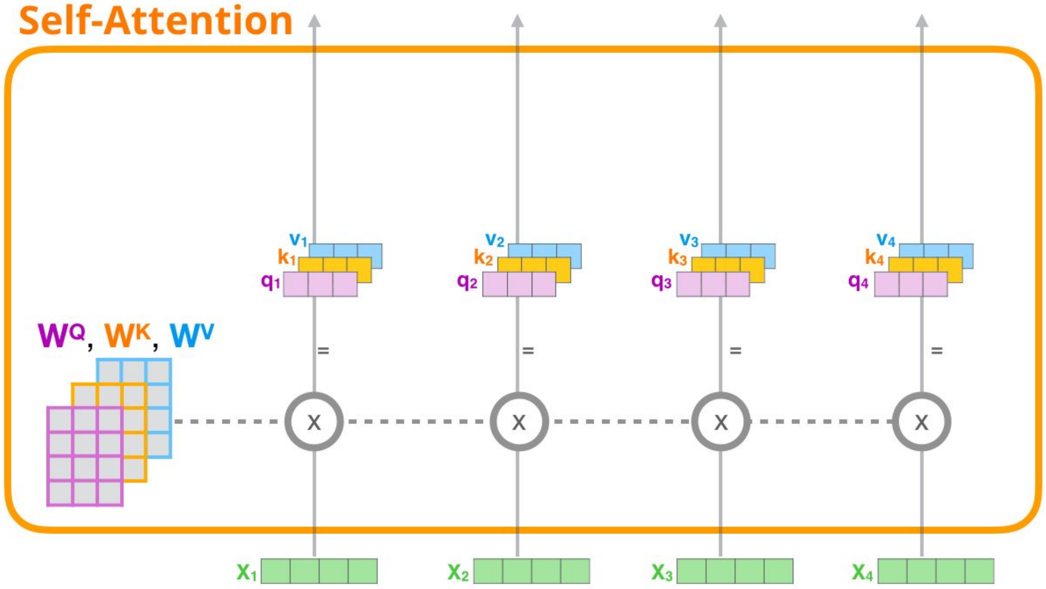
Transformer - self-attention mechanism





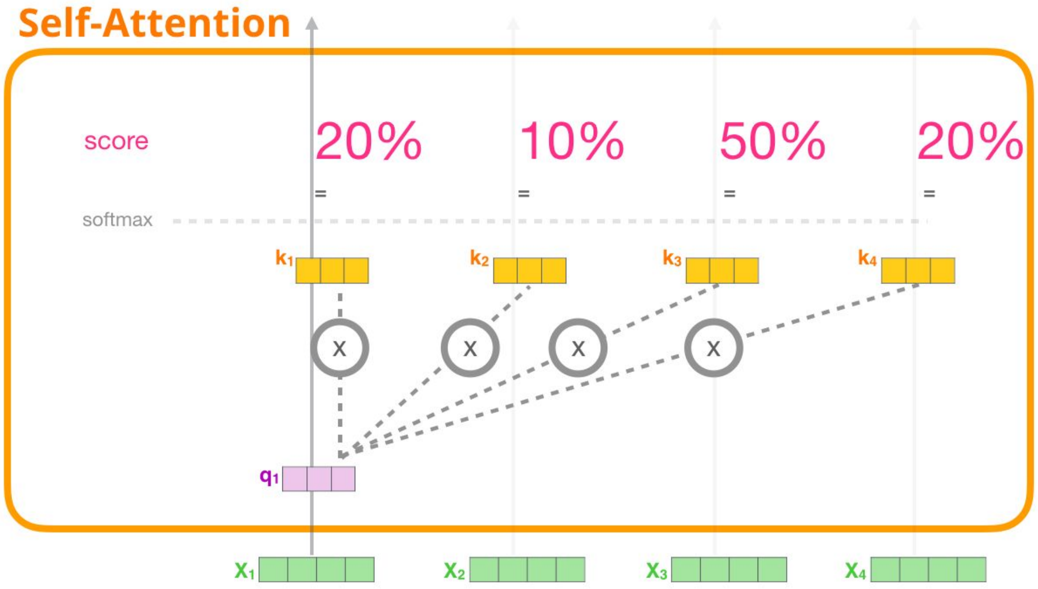
Transformer - self-attention mechanism

1) For each input token, create a query vector, a key vector, and a value vector by multiplying by weight Matrices W^Q, W^K, W^V

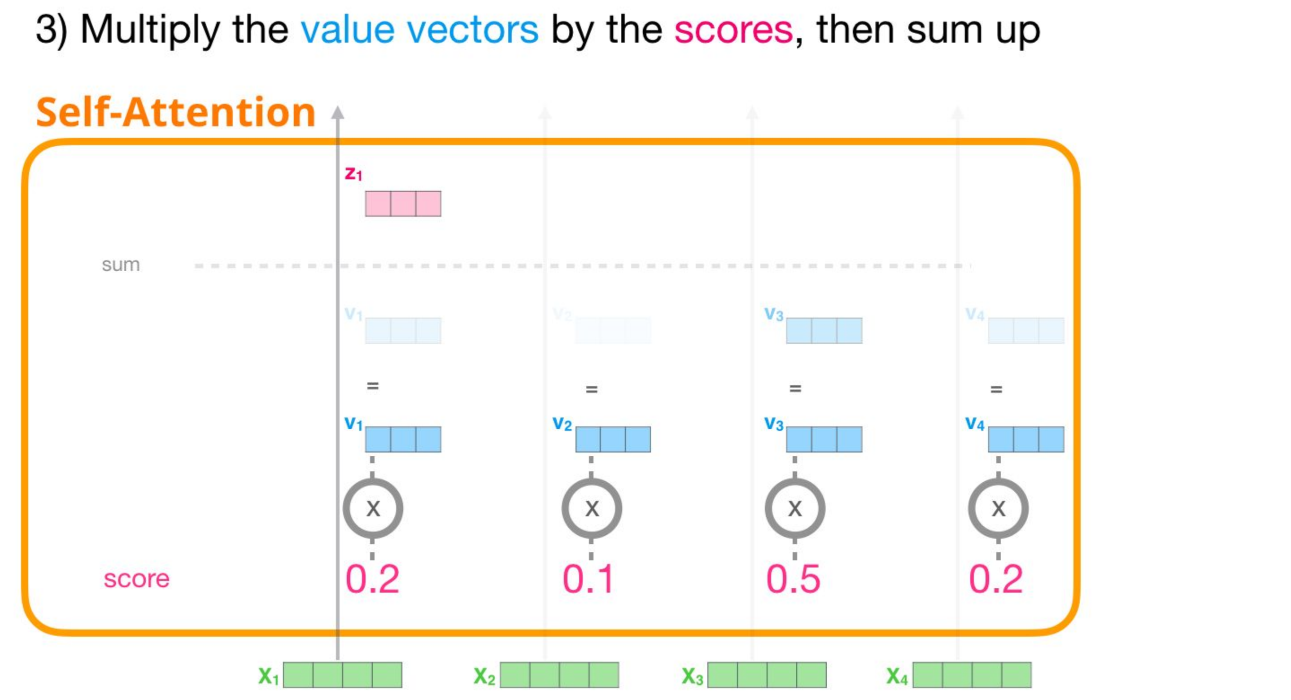


Transformer - self-attention mechanism

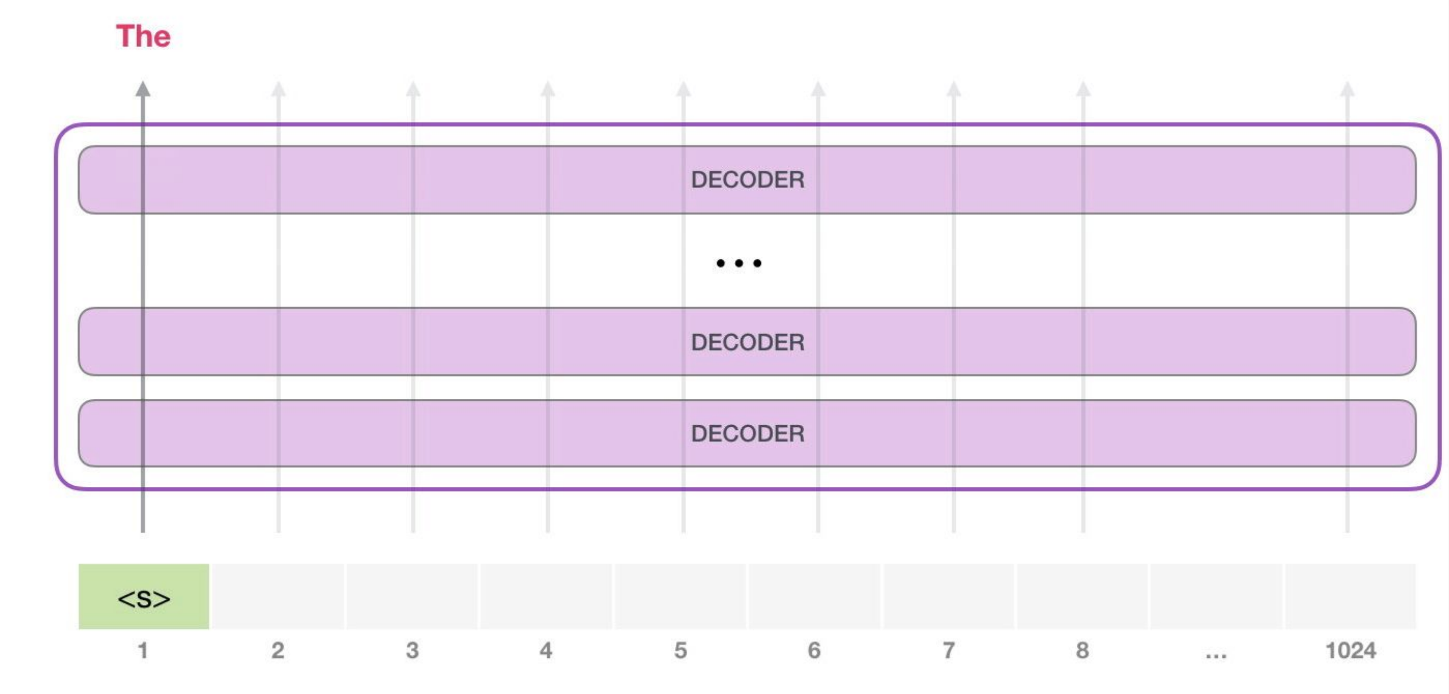
2) Multiply (dot product) the current query vector, by all the key vectors, to get a score of how well they match



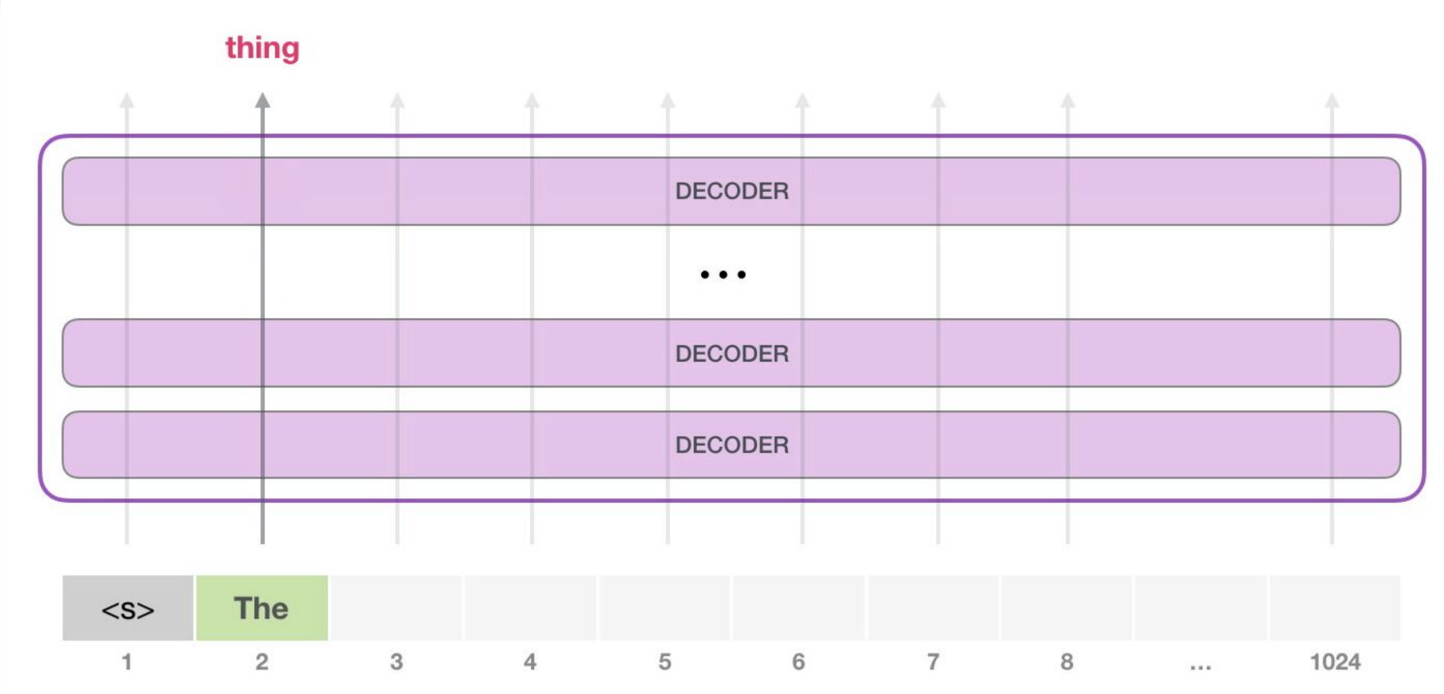
Transformer - self-attention mechanism



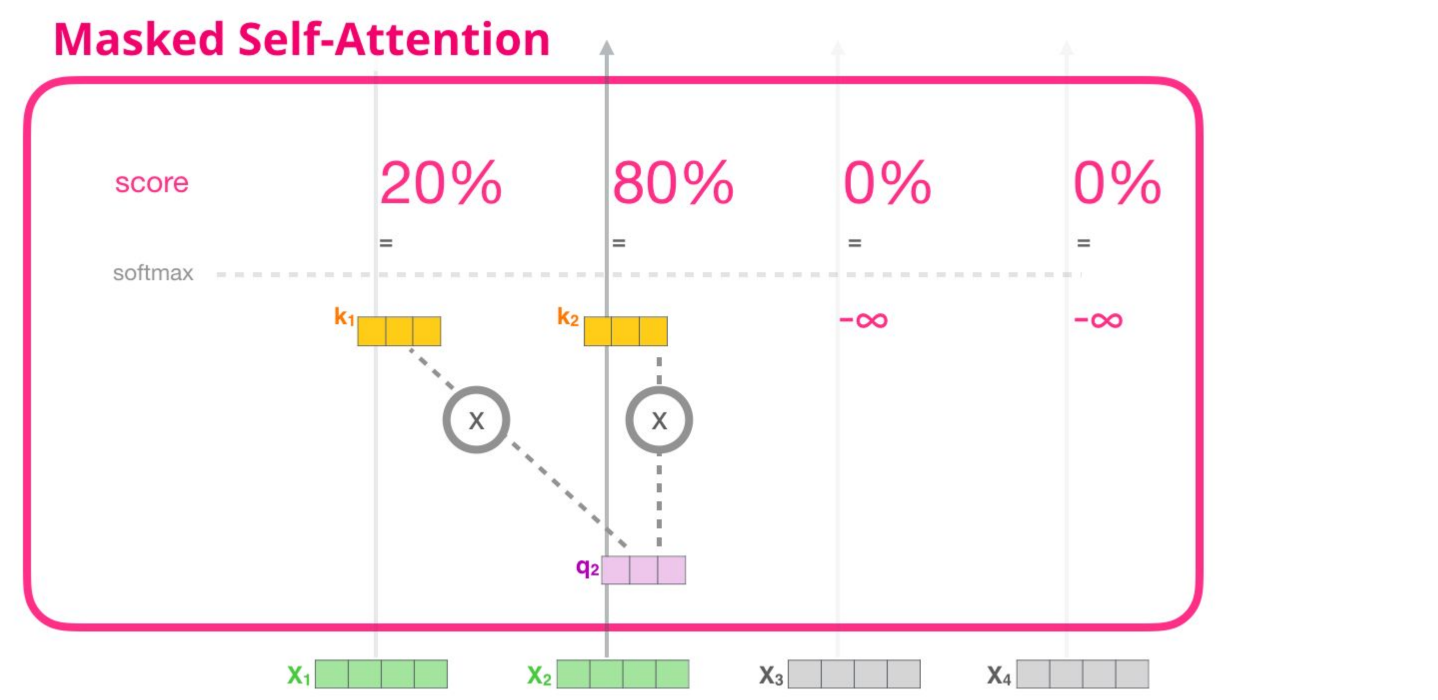
GPT-2 - overview



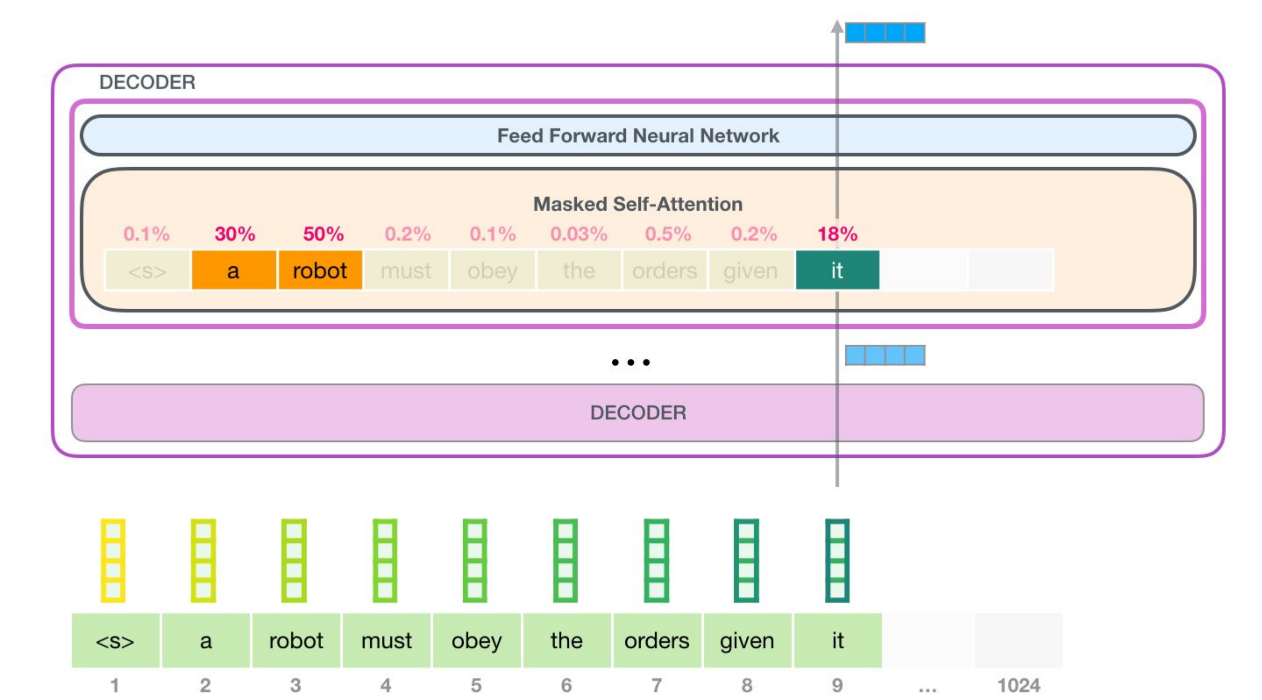
GPT-2 - overview



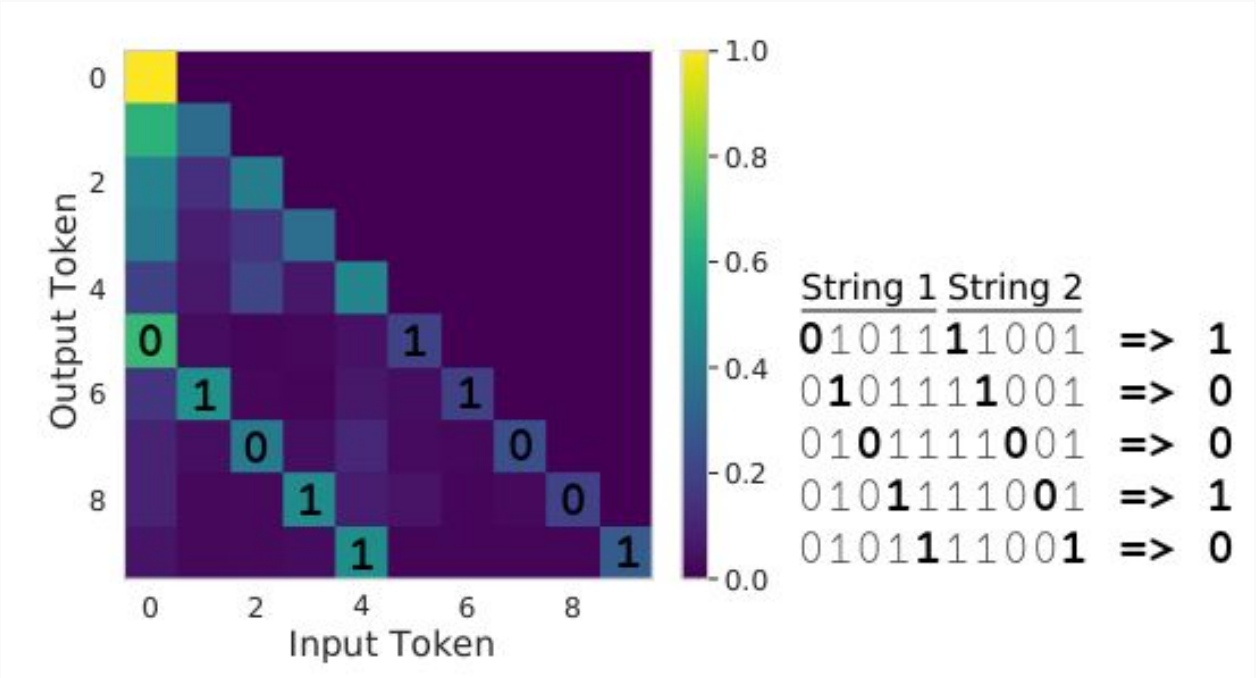
GPT-2 - masked self-attention



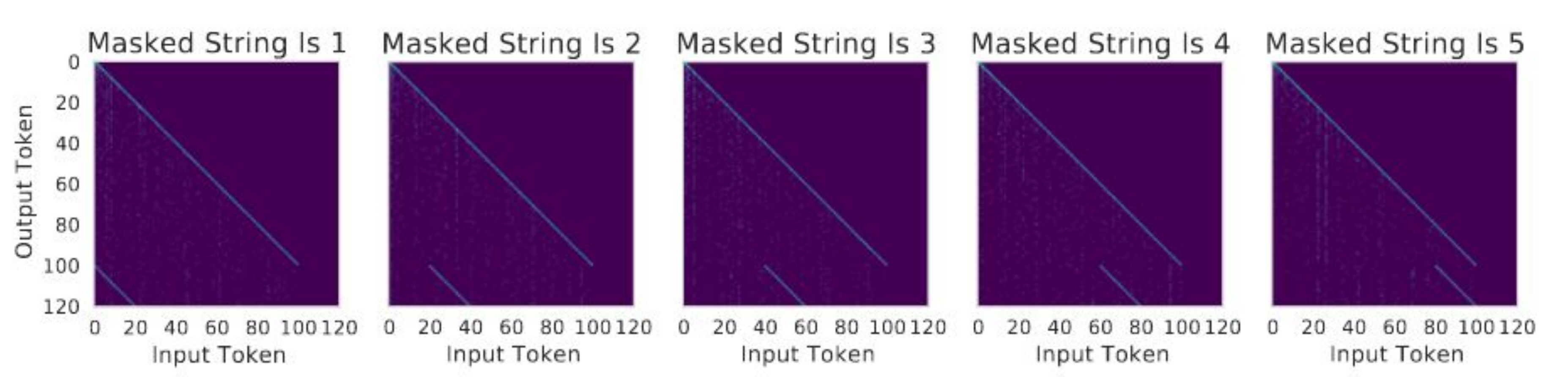
GPT-2 - decoder block



GPT-2 - first layer attention for bit XOR



GPT-2 - first layer attention for bit memory



Option II: the pre-training

Pre-training

Perhaps large corpora of human language data exhibit special properties that make universal computation feasible and thus generalize well to vastly different problems.

Pre-training

Perhaps large corpora of human language data exhibit special properties that make universal computation feasible and thus generalize well to vastly different problems.

Can we identify and extract these properties?

Pre-Training a Language Model Without Human Language

Cheng-Han Chiang, Hung-yi Lee

arXiv:2012.11995

Overview

An earlier paper, performing experiments “in reverse”.

Model is pre-trained on one of few selected tasks, and then finetuned (unfortunately without freezing) on NLP benchmarks.

Overview

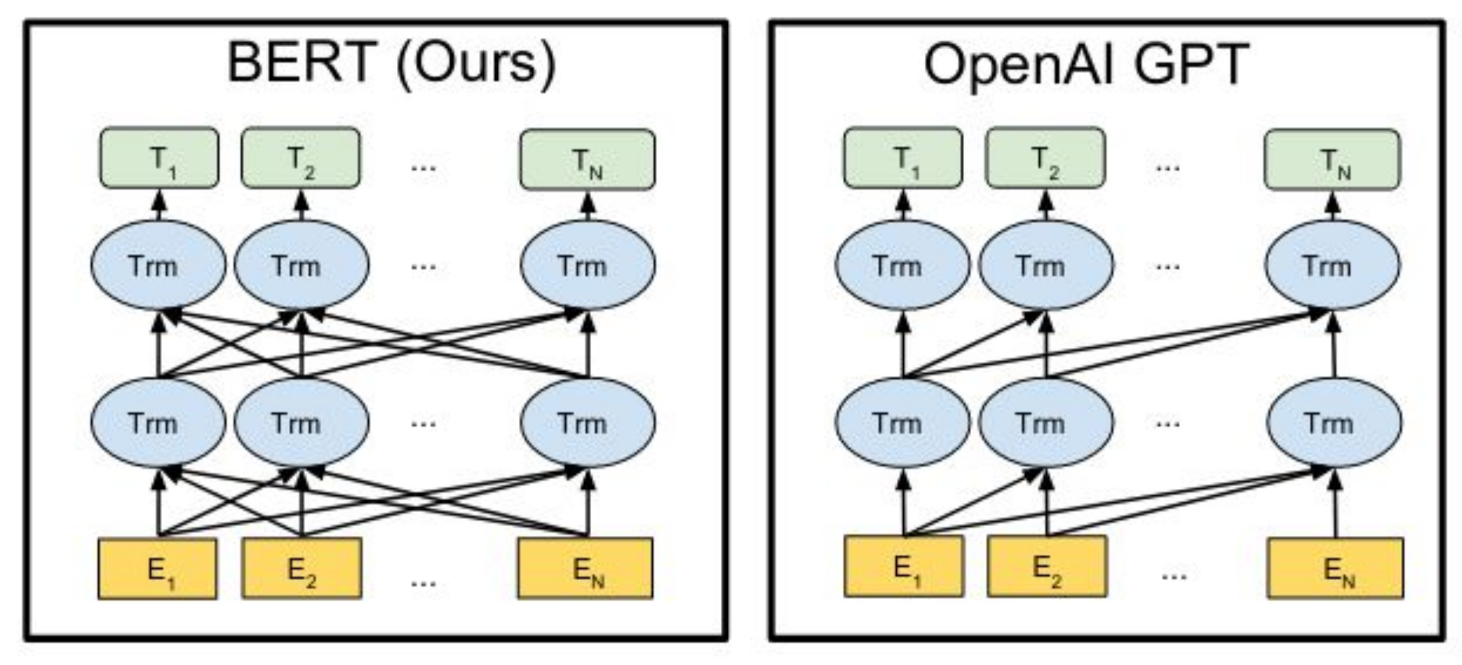
The question posed relates to Universal Computation as follows: what other pre-training tasks allow the resulting model to easily finetune to an NLP task? What are their characteristics?

Model

Instead of GPT-2, the model of choice is RoBERTa.

The model size is similar to GPT-2 (12 layers, hidden dimension 768, 12 attention heads, ~110M parameters).

Model - differences with GPT



Model - differences with GPT

Note that, from the Universal Computation paper:

Task	GPT-2 (FPT Default)	BERT	T5	Longformer
ListOps	38.4%	38.3%	15.4%	17.0%
CIFAR-10	68.2%	68.8%	64.7%	66.8%

Baseline datasets

There are 3 baselines trained on ~80MB of data:

- sequences (90-120 tokens) sampled uniformly from ~30k tokens
- sequences (90-120 tokens) sampled according to English distribution from ~30k tokens
- English wikipedia masked language model

\+ no pre-training (from scratch on downstream tasks)

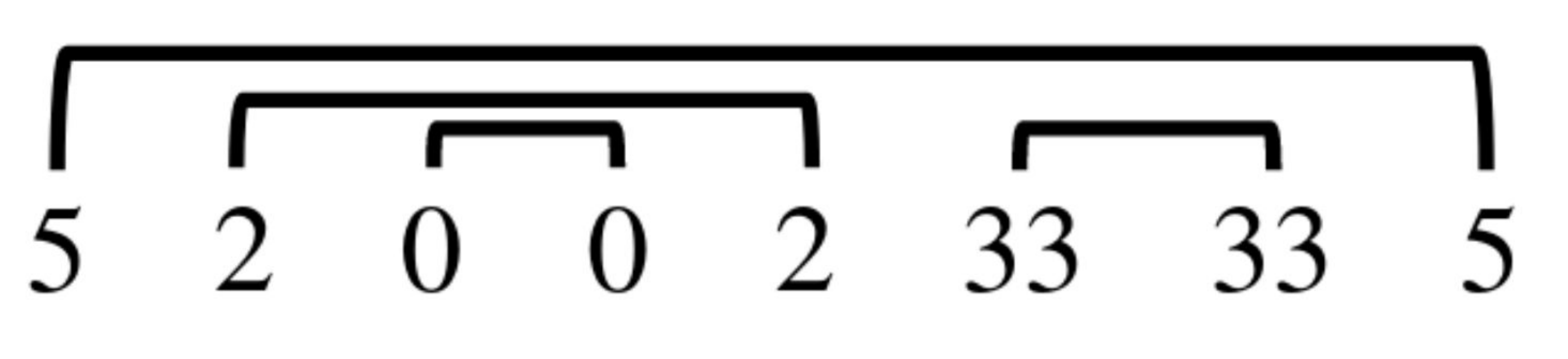
Comparison datasets

- proteins: 14,670,860 sequences, split into 3,150 families
- tokenized JavaScript without comments: 10M tokens total, vocab size is 30k
- stack-based Bernoulli grammar
- esoteric human language: Kannada

Comparison datasets - stack-based grammar

Vocab size is ~30k. For each step, with probability 0.4 a random (English distribution) token is added to the sequence and on top of a stack. Otherwise (0.6) the stack is popped giving the next token. This creates proper parentheses-expressions.

Comparison datasets - stack-based grammar



Comparison datasets - Kannada

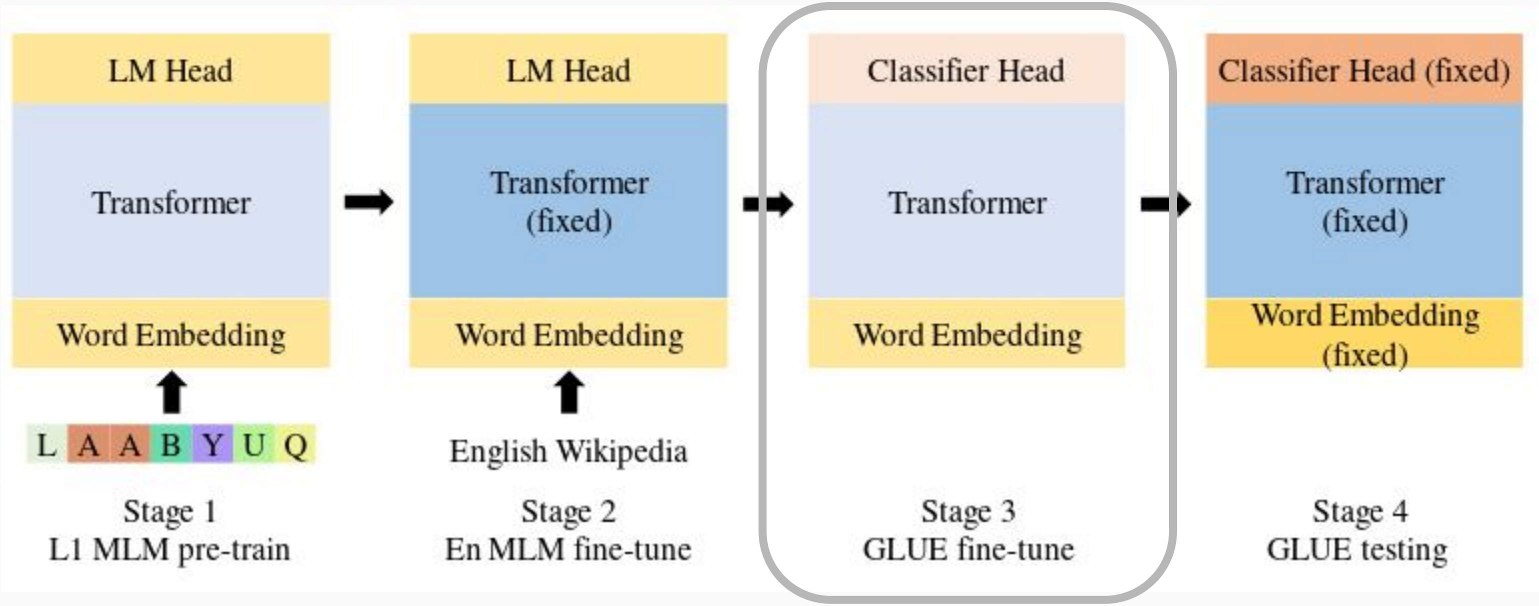
Language from India, 43M + 13M speakers.

Vocab size is 30k. Sentence structure is SOV (contrasted to English SVO). The language was chosen to differ significantly from target language.

Training and benchmarks

The models are trained like the original BERT model, and benchmarks also follow the GLUE benchmarks like BERT.

Training


LM Head
Transformer
Word Embedding
L A A B Y U Q
Stage 1
L1 MLM pre-train
LM Head
Transformer (fixed)
Word Embedding
English Wikipedia
Stage 2
En MLM fine-tune
Classifier Head
Transformer
Word Embedding
Stage 3
GLUE fine-tune
Classifier Head (fixed)
Transformer (fixed)
Word Embedding (fixed)
Stage 4
GLUE testing

Results

	L1	STS-B	QNLI	QQP	CoLA	SST-2	MNLI	MRPC	RTE	Avg
1	**No Pre-train**	0.17	0.60	0.75	0.13	0.83	0.64	0.67	0.50	0.54
	Pre-train En	0.76	0.83	0.86	0.34	0.88	0.76	0.77	0.53	0.72
2	**Rand. Baseline**	0.29	0.66	0.80	0.14	0.83	0.65	0.77	0.51	0.58
	Zipf Baseline	0.38	0.66	0.80	0.11	0.82	0.64	0.81	0.49	0.59
3	**Amino Acid**	0.16	0.65	0.79	0.07	0.82	0.60	0.81	0.44	0.55
	Java Script	0.31	0.67	0.77	0.02	0.81	0.66	0.75	0.51	0.56
	Kannada	0.76	0.77	0.83	0.12	0.81	0.69	0.80	0.55	0.67
4	**Artificial (Uni.)**	0.72	0.77	0.80	0.14	0.81	0.69	0.77	0.52	0.65
	Artificial (Zipf)	0.76	0.77	0.83	0.11	0.82	0.69	0.75	0.53	0.66
5	**Artificial (5000)**	0.73	0.76	0.82	0.09	0.84	0.69	0.82	0.53	0.66
	Artificial (500)	0.42	0.68	0.80	0.08	0.81	0.68	0.79	0.51	0.60
	Artificial (50)	0.18	0.62	0.74	0.06	0.82	0.61	0.77	0.52	0.54
	Artificial (50-s)	0.65	0.73	0.84	0.06	0.80	0.64	0.75	0.50	0.62

Strong alternative datasets

	L1	STS-B	QNLI	QQP	CoLA	SST-2	MNLI	MRPC	RTE	Avg
1	**No Pre-train**	0.17	0.60	0.75	0.13	0.83	0.64	0.67	0.50	0.54
	Pre-train En	0.76	0.83	0.86	0.34	0.88	0.76	0.77	0.53	0.72
2	**Rand. Baseline**	0.29	0.66	0.80	0.14	0.83	0.65	0.77	0.51	0.58
	Zipf Baseline	0.38	0.66	0.80	0.11	0.82	0.64	0.81	0.49	0.59
3	**Amino Acid**	0.16	0.65	0.79	0.07	0.82	0.60	0.81	0.44	0.55
	Java Script	0.31	0.67	0.77	0.02	0.81	0.66	0.75	0.51	0.56
	Kannada	0.76	0.77	0.83	0.12	0.81	0.69	0.80	0.55	0.67
4	**Artificial (Uni.)**	0.72	0.77	0.80	0.14	0.81	0.69	0.77	0.52	0.65
	Artificial (Zipf)	0.76	0.77	0.83	0.11	0.82	0.69	0.75	0.53	0.66
5	**Artificial (5000)**	0.73	0.76	0.82	0.09	0.84	0.69	0.82	0.53	0.66
	Artificial (500)	0.42	0.68	0.80	0.08	0.81	0.68	0.79	0.51	0.60
	Artificial (50)	0.18	0.62	0.74	0.06	0.82	0.61	0.77	0.52	0.54
	Artificial (50-s)	0.65	0.73	0.84	0.06	0.80	0.64	0.75	0.50	0.62

Weak alternative datasets

	L1	STS-B	QNLI	QQP	CoLA	SST-2	MNLI	MRPC	RTE	Avg
1	**No Pre-train**	0.17	0.60	0.75	0.13	0.83	0.64	0.67	0.50	0.54
	Pre-train En	0.76	0.83	0.86	0.34	0.88	0.76	0.77	0.53	0.72
2	**Rand. Baseline**	0.29	0.66	0.80	0.14	0.83	0.65	0.77	0.51	0.58
	Zipf Baseline	0.38	0.66	0.80	0.11	0.82	0.64	0.81	0.49	0.59
3	**Amino Acid**	0.16	0.65	0.79	0.07	0.82	0.60	0.81	0.44	0.55
	Java Script	0.31	0.67	0.77	0.02	0.81	0.66	0.75	0.51	0.56
	Kannada	0.76	0.77	0.83	0.12	0.81	0.69	0.80	0.55	0.67
4	**Artificial (Uni.)**	0.72	0.77	0.80	0.14	0.81	0.69	0.77	0.52	0.65
	Artificial (Zipf)	0.76	0.77	0.83	0.11	0.82	0.69	0.75	0.53	0.66
5	**Artificial (5000)**	0.73	0.76	0.82	0.09	0.84	0.69	0.82	0.53	0.66
	Artificial (500)	0.42	0.68	0.80	0.08	0.81	0.68	0.79	0.51	0.60
	Artificial (50)	0.18	0.62	0.74	0.06	0.82	0.61	0.77	0.52	0.54
	Artificial (50-s)	0.65	0.73	0.84	0.06	0.80	0.64	0.75	0.50	0.62

Importance of distribution for unstructured data

	L1	STS-B	QNLI	QQP	CoLA	SST-2	MNLI	MRPC	RTE	Avg
1	**No Pre-train**	0.17	0.60	0.75	0.13	0.83	0.64	0.67	0.50	0.54
	Pre-train En	0.76	0.83	0.86	0.34	0.88	0.76	0.77	0.53	0.72
2	**Rand. Baseline**	0.29	0.66	0.80	0.14	0.83	0.65	0.77	0.51	0.58
	Zipf Baseline	0.38	0.66	0.80	0.11	0.82	0.64	0.81	0.49	0.59
3	**Amino Acid**	0.16	0.65	0.79	0.07	0.82	0.60	0.81	0.44	0.55
	Java Script	0.31	0.67	0.77	0.02	0.81	0.66	0.75	0.51	0.56
	Kannada	0.76	0.77	0.83	0.12	0.81	0.69	0.80	0.55	0.67
4	**Artificial (Uni.)**	0.72	0.77	0.80	0.14	0.81	0.69	0.77	0.52	0.65
	Artificial (Zipf)	0.76	0.77	0.83	0.11	0.82	0.69	0.75	0.53	0.66
5	**Artificial (5000)**	0.73	0.76	0.82	0.09	0.84	0.69	0.82	0.53	0.66
	Artificial (500)	0.42	0.68	0.80	0.08	0.81	0.68	0.79	0.51	0.60
	Artificial (50)	0.18	0.62	0.74	0.06	0.82	0.61	0.77	0.52	0.54
	Artificial (50-s)	0.65	0.73	0.84	0.06	0.80	0.64	0.75	0.50	0.62

Importance of distribution for structured data

	L1	STS-B	QNLI	QQP	CoLA	SST-2	MNLI	MRPC	RTE	Avg
1	No Pre-train	0.17	0.60	0.75	0.13	0.83	0.64	0.67	0.50	0.54
	Pre-train En	0.76	0.83	0.86	0.34	0.88	0.76	0.77	0.53	0.72
2	Rand. Baseline	0.29	0.66	0.80	0.14	0.83	0.65	0.77	0.51	0.58
	Zipf Baseline	0.38	0.66	0.80	0.11	0.82	0.64	0.81	0.49	0.59
3	Amino Acid	0.16	0.65	0.79	0.07	0.82	0.60	0.81	0.44	0.55
	Java Script	0.31	0.67	0.77	0.02	0.81	0.66	0.75	0.51	0.56
	Kannada	0.76	0.77	0.83	0.12	0.81	0.69	0.80	0.55	0.67
4	Artificial (Uni.)	0.72	0.77	0.80	0.14	0.81	0.69	0.77	0.52	0.65
	Artificial (Zipf)	0.76	0.77	0.83	0.11	0.82	0.69	0.75	0.53	0.66
5	Artificial (5000)	0.73	0.76	0.82	0.09	0.84	0.69	0.82	0.53	0.66
	Artificial (500)	0.42	0.68	0.80	0.08	0.81	0.68	0.79	0.51	0.60
	Artificial (50)	0.18	0.62	0.74	0.06	0.82	0.61	0.77	0.52	0.54
	Artificial (50-s)	0.65	0.73	0.84	0.06	0.80	0.64	0.75	0.50	0.62

Importance of token number mismatch

	L1	STS-B	QNLI	QQP	CoLA	SST-2	MNLI	MRPC	RTE	Avg
1	**No Pre-train**	0.17	0.60	0.75	0.13	0.83	0.64	0.67	0.50	0.54
	Pre-train En	0.76	0.83	0.86	0.34	0.88	0.76	0.77	0.53	0.72
2	**Rand. Baseline**	0.29	0.66	0.80	0.14	0.83	0.65	0.77	0.51	0.58
	Zipf Baseline	0.38	0.66	0.80	0.11	0.82	0.64	0.81	0.49	0.59
3	**Amino Acid**	0.16	0.65	0.79	0.07	0.82	0.60	0.81	0.44	0.55
	Java Script	0.31	0.67	0.77	0.02	0.81	0.66	0.75	0.51	0.56
	Kannada	0.76	0.77	0.83	0.12	0.81	0.69	0.80	0.55	0.67
4	**Artificial (Uni.)**	0.72	0.77	0.80	0.14	0.81	0.69	0.77	0.52	0.65
	Artificial (Zipf)	0.76	0.77	0.83	0.11	0.82	0.69	0.75	0.53	0.66
5	**Artificial (5000)**	0.73	0.76	0.82	0.09	0.84	0.69	0.82	0.53	0.66
	Artificial (500)	0.42	0.68	0.80	0.08	0.81	0.68	0.79	0.51	0.60
	Artificial (50)	0.18	0.62	0.74	0.06	0.82	0.61	0.77	0.52	0.54
	Artificial (50-s)	0.65	0.73	0.84	0.06	0.80	0.64	0.75	0.50	0.62

Token number mismatch with substitution

	L1	STS-B	QNLI	QQP	CoLA	SST-2	MNLI	MRPC	RTE	Avg
1	**No Pre-train**	0.17	0.60	0.75	0.13	0.83	0.64	0.67	0.50	0.54
	Pre-train En	0.76	0.83	0.86	0.34	0.88	0.76	0.77	0.53	0.72
2	**Rand. Baseline**	0.29	0.66	0.80	0.14	0.83	0.65	0.77	0.51	0.58
	Zipf Baseline	0.38	0.66	0.80	0.11	0.82	0.64	0.81	0.49	0.59
3	**Amino Acid**	0.16	0.65	0.79	0.07	0.82	0.60	0.81	0.44	0.55
	Java Script	0.31	0.67	0.77	0.02	0.81	0.66	0.75	0.51	0.56
	Kannada	0.76	0.77	0.83	0.12	0.81	0.69	0.80	0.55	0.67
4	**Artificial (Uni.)**	0.72	0.77	0.80	0.14	0.81	0.69	0.77	0.52	0.65
	Artificial (Zipf)	0.76	0.77	0.83	0.11	0.82	0.69	0.75	0.53	0.66
5	**Artificial (5000)**	0.73	0.76	0.82	0.09	0.84	0.69	0.82	0.53	0.66
	Artificial (500)	0.42	0.68	0.80	0.08	0.81	0.68	0.79	0.51	0.60
	Artificial (50)	0.18	0.62	0.74	0.06	0.82	0.61	0.77	0.52	0.54
	Artificial (50-s)	0.65	0.73	0.84	0.06	0.80	0.64	0.75	0.50	0.62

Back to our question!

The question

Why do these models generalize to non-NLP tasks?

Architectural factors (e.g. attention as generalization of MLP, wide context with long range, multiple independent learned “tools”, “blackboard” model of brain etc.)

Pre-training factors (e.g. difficult task forces general methods, long-range dependencies as most important natural signals, large datasets allow less bias in architecture etc.)

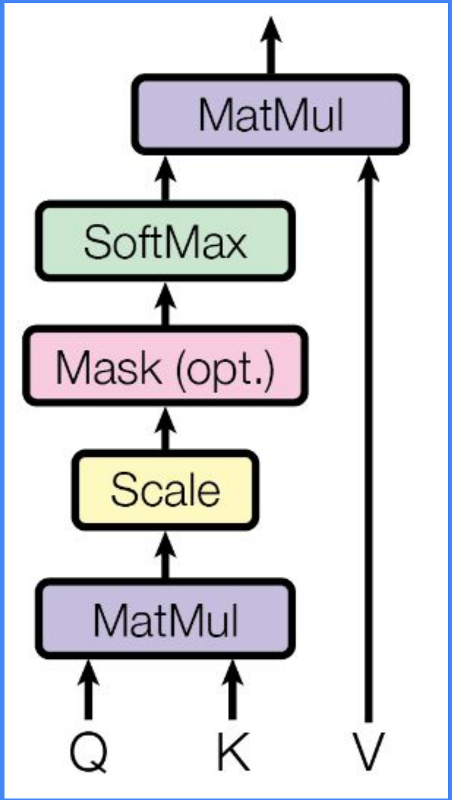
Example follow-ups

Is some synthetic dataset better than language / images?

Is the Universal Computation capability unique to Transformers?

Can a synthetic dataset allow for orders of magnitude more data and so enable even more general models?

Can multiple datasets be combined for better pre-training?

Thank you for your 

!