Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J. Liu

Presentation by: Rafael Ferreira

Date: 14/05/2020

Content

T5 – Text-to-Text Transfer Transformer

Text-To-Text – Relates to the input and output formats of the model.

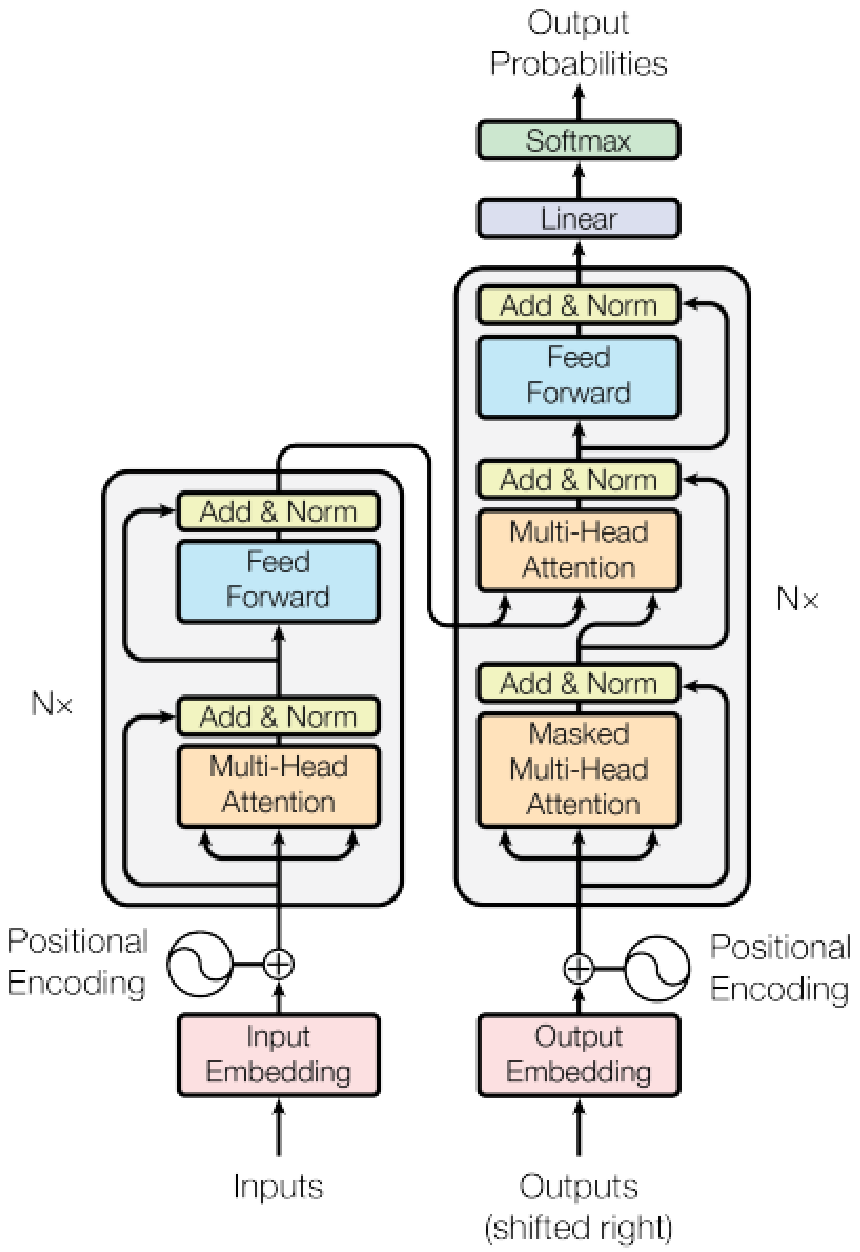
The model receives input to utilize as context to then produce an output.

Transfer – Transfer learning. A Transfer learning model is a model that is first pre-trained on large amounts of data (unsupervised) and is then fine-tuned on a task.

Transformer – The architecture of the model.

The Transformer

- The encoder-decoder version of T5 follows the architecture proposed in **Attention is all you need**[1].
- One of the most important parts of this architecture is the **self-attention** mechanism. A variant of attention that replaces each element in a sequence by a weighted average of the rest of the sequence.
- A tutorial on the attention mechanism and the transformer architecture can be seen [here].

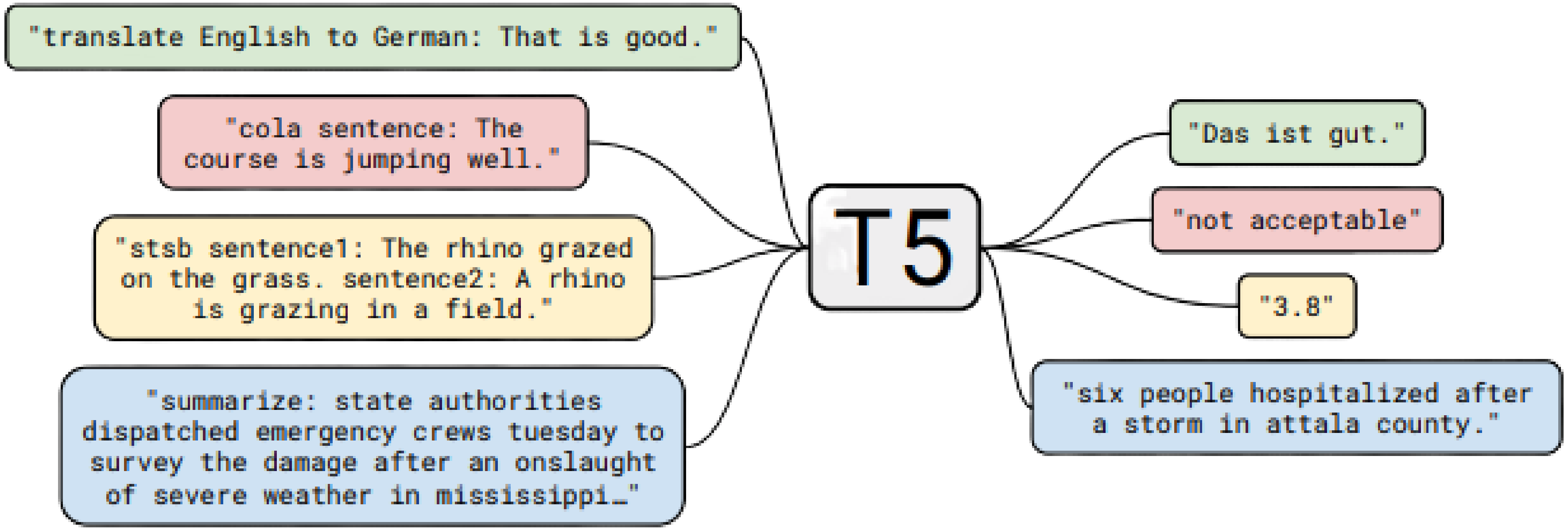


[1]Ashish Vaswani et al., "Attention Is All You Need"

T5 Applications

One of the advantages of a text-to-text format is that it provides a **consistent training objective** both for pre-training and fine-tuning.
T5 is also able to handle **multiple tasks** by adding task-specific prefixes to the original input sequence.

- Machine Translation
- Classification
- “Regression”
- Question Answering
- Summarization
- Coreference Resolution



Colin Raffel et al. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”

Tasks and Datasets

- **GLUE**[1] and **SuperGLUE**[2] – collection of text classification tasks.
- **CNNDM**[3] (CNN/Daily Mail) – summarization task.
- **SQuAD**[4] – extractive question-answering dataset.
- **EnDe**, **EnFr** and **EnRo** – translation tasks from English to German (Deutsch), French, and Romanian respectively.

[1]Wang et al., "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding"
[2]Wang et al., "SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems"
[3]Hermann et al., "Teaching Machines to Read and Comprehend"
[4]Rajpurkar et al, "Squad: 100,000+ questions for machine comprehension of text"

T5 Model - Baseline

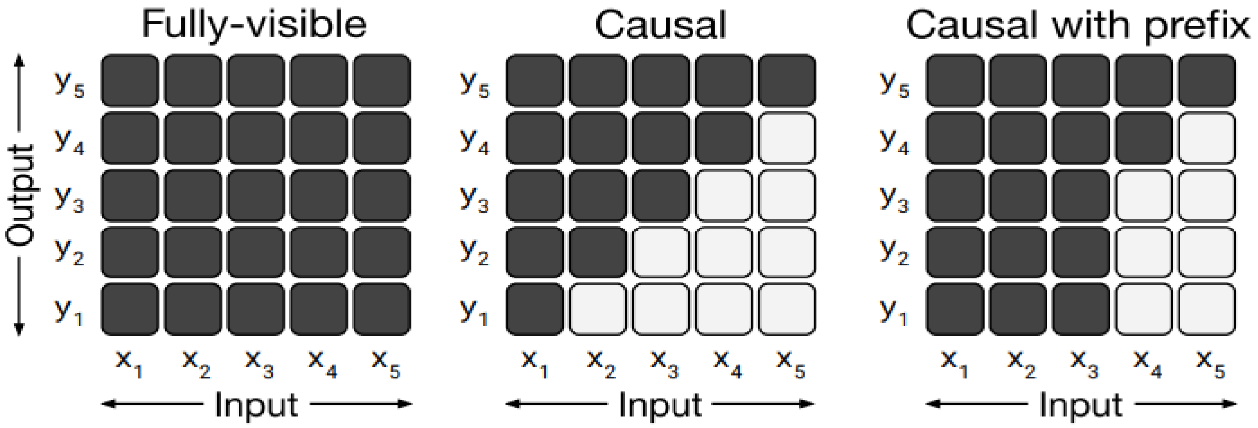
- **Architecture** - Encoder-decoder transformer architecture.
  - 24 layers / hidden dimension size 768 / 12 attention heads,
  - 220 Million parameters (similar to $\text{BERT}_{\text{BASE}}$).
- **Training** - Maximum likelihood.
  - Pre-trained model for 2^{19} steps and fine-tuned for 2^{18} steps,
  - Maximum sequence length of 512 tokens and batch size of 128 sequences.

- **Unsupervised Objective** - "denoising", predicting missing or corrupted tokens from the input.

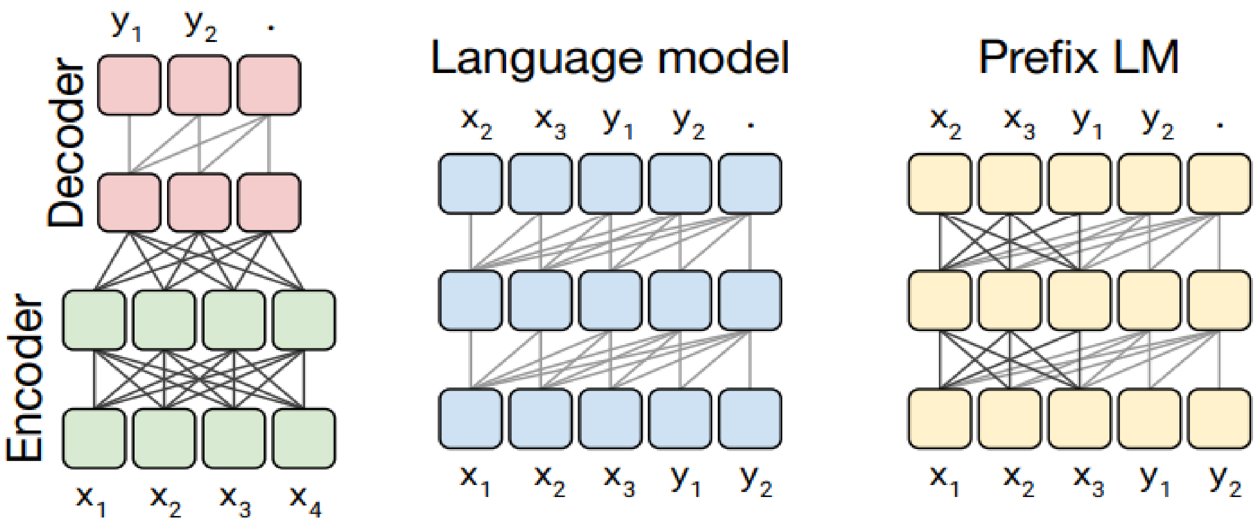
	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline average	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Baseline standard deviation	0.235	0.065	0.343	0.416	0.112	0.090	0.108
No pre-training	66.22	17.60	50.31	53.04	25.86	**39.77**	24.04

Architectures

- **Attention Masks**
  - Used to zero out certain weights to attend only to inputs it considers necessary at each step.



- **Model Structures:**
  - **Encoder-decoder:**
    - Encoder uses fully-visible attention mask
    - Decoder uses causal attention mask
  - **Language model:**
    - Next Step Prediction
  - **Prefix LM**
    - Fully-visible attention mask on prefix
    - Causal attention mask outside prefix



Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer"

Architectures

- Considering the number of layers and parameters as **L** and **P** respectively of a $\text{BERT}_{\text{BASE}}$ model, and **M** as the number of FLOPs.

	Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★	Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
	Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
	Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
	Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
	Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39

Unsupervised Objectives

- Decision of which algorithm to use as the unsupervised objective during pre-training is essential, because it provides the method through which the model **“learns the knowledge”**.

- Examples of unsupervised objectives for the sentence:
  - “Thank you for inviting me to your party last week.”

Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style	Thank you <M> <M> me to your party apple week .	*(original text)*
Deshuffling	party me for your to . last fun you inviting week Thank	*(original text)*
I.i.d. noise, mask tokens	Thank you <M> <M> me to your party <M> week .	*(original text)*
I.i.d. noise, replace spans	Thank you <X> me to your party <Y> week .	<X> for inviting <Y> last <Z>
I.i.d. noise, drop tokens	Thank you me to your party week .	for inviting last
Random spans	Thank you <X> to <Y> week .	<X> for inviting me <Y> your party last <Z>

Unsupervised Objectives

- Deshuffling objectives perform significantly worse.
- Denoising objectives are the best but comparable between themselves.
- The major advantage of span corruption comes from their **lower computational cost**, because on average, span corruption produces shorter sequences.

Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Prefix language modeling	80.69	18.94	77.99	65.27	**26.86**	39.73	**27.49**
BERT-style [Devlin et al., 2018]	**82.96**	**19.17**	**80.65**	**69.85**	**26.78**	**40.03**	**27.41**
Deshuffling	73.17	18.59	67.61	58.47	26.11	39.30	25.62
MASS-style [Song et al., 2019]	82.32	19.16	80.10	69.28	26.79	**39.89**	27.55
★ Replace corrupted spans	83.28	**19.24**	**80.88**	**71.36**	**26.98**	39.82	**27.65**
Drop corrupted tokens	**84.44**	**19.31**	**80.52**	68.67	**27.07**	39.76	**27.82**

Pre-Training Datasets

- The pre-training dataset is another essential part of training, alongside the pre-training objective.
- There exists various large-scale unlabeled datasets. More data being produced each month.

- C4
- RealNews-like (C4 news)
- WebText-Like (C4 Reddit)
- Wikipedia
- Toronto Book Corpus

Dataset	Size	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ C4	745GB	83.28	**19.24**	80.88	71.36	**26.98**	**39.82**	**27.65**
C4, unfiltered	6.1TB	81.46	19.14	78.78	68.04	26.55	39.34	27.21
RealNews-like	35GB	**83.83**	**19.23**	80.39	72.38	**26.75**	**39.90**	**27.48**
WebText-like	17GB	**84.03**	**19.31**	**81.42**	71.40	**26.80**	**39.74**	**27.59**
Wikipedia	16GB	81.85	**19.31**	81.29	68.01	**26.94**	39.69	**27.67**
Wikipedia + TBC	20GB	83.65	**19.28**	**82.08**	**73.24**	**26.77**	39.63	**27.57**

- Main takeaway is that **pre-training on in-domain** unlabeled data can improve performance on downstream tasks.

Pre-Training Datasets

- **Influence of repeating data** - Tested using the C4 dataset truncated at different sizes.

Number of tokens	Repeats	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Full dataset	0	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
2^{29}	64	**82.87**	**19.19**	**80.97**	**72.03**	**26.83**	**39.74**	**27.63**
2^{27}	256	82.62	**19.20**	79.78	69.97	**27.02**	**39.71**	27.33
2^{25}	1,024	79.55	18.57	76.27	64.76	26.38	39.56	26.80
2^{23}	4,096	76.34	18.33	70.92	59.29	26.37	38.84	25.81

- **Performance** generally **degrades** as the **size** of the dataset **decreases**.
- When the dataset is repeated 64 times it surpasses the full dataset in some tasks showing that some amount of repetition might not be harmful.

Training strategies

- **Fine-tuning methods**
  - **Adapter Layers**[1] – dense ReLU blocks added after each existing FFN in each block of the Transformer. Only the **adapter layers and layer normalization parameters are updated**. d is the inner dimensionality of the feed-forward network, which changes the number of parameters of the model.
  - **Gradual Unfreezing**[2] – more parameters finetuned over time. The “unfreezing” starts at the top layer.

Fine-tuning method	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ All parameters	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Adapter layers, $d = 32$	80.52	15.08	79.32	60.40	13.84	17.88	15.54
Adapter layers, $d = 128$	81.51	16.62	79.47	63.03	19.83	27.50	22.63
Adapter layers, $d = 512$	81.54	17.78	79.18	64.30	23.45	33.98	25.81
Adapter layers, $d = 2048$	81.51	16.62	79.47	63.03	19.83	27.50	22.63
Gradual unfreezing	82.50	18.95	79.17	**70.79**	26.71	39.02	26.93

[1]Houlsby et al. “Parameter-efficient transfer learning for NLP”
[2]Universal language model fine-tuning for text classification

Training strategies

- **Multi-task learning** – train the model at multiple tasks at a time. In T5 this only corresponds to mixing datasets.
- Multiple ways of mixing the datasets:
  - **Proportional mixing** – proportional to the size of each dataset but using an artificial limit K on the size of each dataset.
  - **Temperature-scaled mixing** – using a “temperature” T to control the mixing rate
  - **Equal mixing** – equal probability to every dataset (probably not a good idea)

Mixing strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline (pre-train/fine-tine)	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Equal	76.13	19.02	76.51	63.37	23.89	34.31	26.78
Examples-proportional, $K = 2^{16}$	80.45	19.04	77.25	69.95	24.35	34.99	27.10
Examples-proportional, $K = 2^{17}$	81.56	19.12	77.00	67.91	24.36	35.00	27.25
Examples-proportional, $K = 2^{18}$	81.67	19.07	78.17	67.94	24.57	35.19	27.39
Examples-proportional, $K = 2^{19}$	81.42	**19.24**	79.78	67.30	25.21	36.30	**27.76**
Examples-proportional, $K = 2^{20}$	80.80	**19.24**	**80.36**	67.38	25.66	36.93	**27.68**
Examples-proportional, $K = 2^{21}$	79.83	18.79	79.50	65.10	25.82	37.22	27.13
Temperature-scaled, $T = 2$	81.90	**19.28**	79.42	69.92	25.42	36.72	27.20
Temperature-scaled, $T = 4$	80.56	**19.22**	77.99	69.54	25.04	35.82	27.45
Temperature-scaled, $T = 8$	77.21	19.10	77.14	66.07	24.55	35.35	27.17

Training strategies

- **Combining multi-task learning with finetuning** – model is pre-trained on all tasks but only fine-tuned on some tasks:
  - Pre-train the model on the mixture of datasets and fine-tuning it on each task
  - Pre-train the model on the mixture of datasets and “leave one out” to be fine-tuned on that task.
  - Pre-training on all datasets

Training strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Unsupervised pre-training + fine-tuning	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	39.82	27.65
Multi-task training	81.42	**19.24**	79.78	67.30	25.21	36.30	27.76
Multi-task pre-training + fine-tuning	**83.11**	**19.12**	**80.26**	**71.03**	**27.08**	39.80	**28.07**
Leave-one-out multi-task training	81.98	19.05	79.97	**71.68**	**26.93**	39.79	**27.87**
Supervised multi-task pre-training	79.93	18.96	77.38	65.36	26.81	**40.13**	**28.04**

- Finetuning after pre-training is comparable to the baseline.

- Supervised pre-training performs worse except for the translation tasks

Scaling

- Possibility to scale in various ways:
  - Using a larger model
  - Increasing batch size
  - Training for more steps
  - Ensembling (combination of models)

Scaling strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Baseline	83.28	19.24	80.88	71.36	26.98	39.82	27.65
1× size, 4× training steps	85.33	19.33	82.45	74.72	27.08	40.66	27.93
1× size, 4× batch size	84.60	19.42	82.52	74.64	27.07	40.60	27.84
2× size, 2× training steps	**86.18**	19.66	**84.18**	77.18	27.52	**41.03**	28.19
4× size, 1× training steps	**85.91**	19.73	**83.86**	**78.04**	27.47	40.71	28.10
4× ensembled	84.77	**20.10**	83.09	71.74	**28.05**	40.53	**28.57**
4× ensembled, fine-tune only	84.05	19.57	82.36	71.55	27.55	40.22	28.09

- Increasing both model size and training steps improves the results.
- Increasing only batch size or training steps are equally beneficial.
- Ensembling proves to be a way of improving performance without increasing model size or training time.

Scaling – Model Size

Model Name	# Parameters	Hidden Size	# Attention Heads	# Total Layers
Small	60 Million	512	8	12
Base	220 Million	768	12	24
Large	770 Million	1024	16	24
3B	2.8 Billion	1024	32	48
11B	11 Billion	1024	128	48

11 B = 11.000.000.000 !!!!

Conclusion

- **Text-to-text** – simple and easily understandable format, that can be adapted to various tasks.
- **Architectures** – the best architecture for a text-to-text format is the encoder-decoder, that although having twice as many parameters as "encoder-only" (BERT) it has a similar computational cost.
- **Unsupervised objectives** – denoising objectives performed better. Changes in the typical algorithm can provide more efficient training.
- **Datasets** – performance degrades when a dataset is repeated many times during pre-training.
- **Training strategies** – updating all weights performed best. But there are methods that can perform similarly in terms of results but faster during training, thanks to only updating part of the parameters.
- **Scaling** – various ways of scaling up to improve performance.

Research Opportunities

- **Model are very large** – invest in ways of reducing the size of the existing models or create new cheaper models.
- **More efficient knowledge extraction** – new pre-training objectives that can leverage the text in a more efficient way, not being necessary to use such a large amount of data.
- **Measure similarity** between pre-training and downstream tasks.
- **Language Agnostic models** – develop models that achieve good performance regardless on the text's language.

A Simple Coding Example

- Following this tutorial, we can use Google's free TPU's available in Colab to train a T5 model for QA.
- Basic pipeline for using T5:
  - Use Colab and a Google Cloud Storage Account (300$ free credit)
  - Install the T5 library
  - Pre-process the data
  - Create the necessary datasets (training, validation, test)
  - Create a task or mixture
  - Define model
  - Finetune on the task
  - Save model
  - Evaluate
- You can also play a trivia question game with the model here.

Thank You!