Quebec
Artificial
Intelligence
Institute

Mila

Natural Language Processing Updates
Week 4

Mirko Bronzi
mirko.bronzi@mila.quebec

Spoiler

What happened after the Transformer model?

What happened after BERT?

Where should I look at to start using pre-trained Transformers models (like BERT)?

Transformers

https://unsplash.com/photos/liAwyJ64wHE,https://www.flickr.com/photos/36784173@N08/3769944000,https://github.com/huggingface/transformers

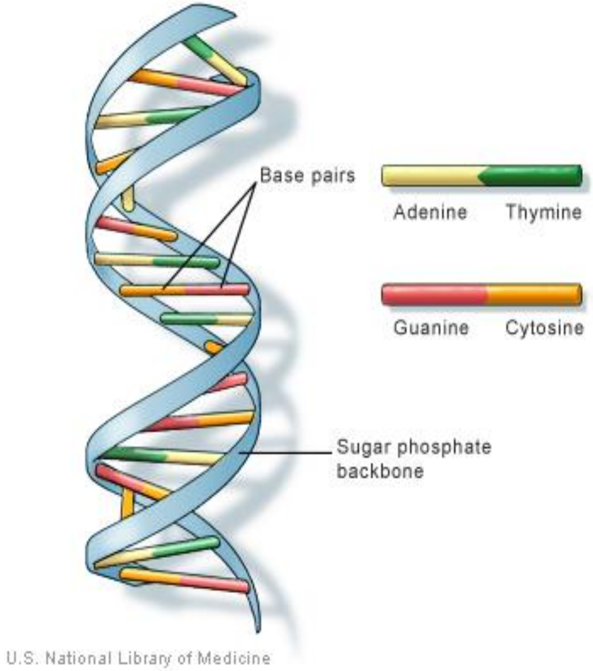
Plan

- **Life after Transformers**
- Life after BERT
- Tools for NLP

What happened after the Transformer model?

Why Are We Interested In Sequences?

- Audio:
  - Speech recognition
  - Text to speech
- Video:
  - Caption generation
  - Movement detection
- Text:
  - Email classification
  - Machine translation
- Medical and Biological data:
  - DNA study
  - Electrocardiogram
- Time series (stocks, weather, ...)
- Etc…



There is a lot of data with sequences!

Short Sequences: Simple RNN

S	e	q	u	e	n	c	e		l	e	n	g	t	h

- Limited to short sequences because of the vanishing gradient problem.

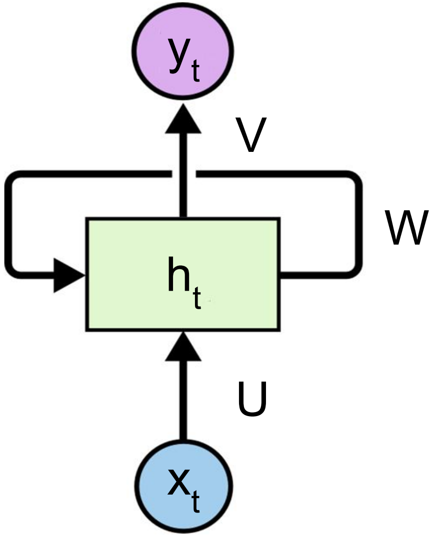


Image from Christopher Olah's blog

Longer Sequences: LSTM

S	e	q	u	e	n	c	e		l	e	n	g	t	h

- Greatly reduces the vanishing gradient problem.
- Still:
    - Hard to parallelize.
    - Information does not flow easily across long sequences.

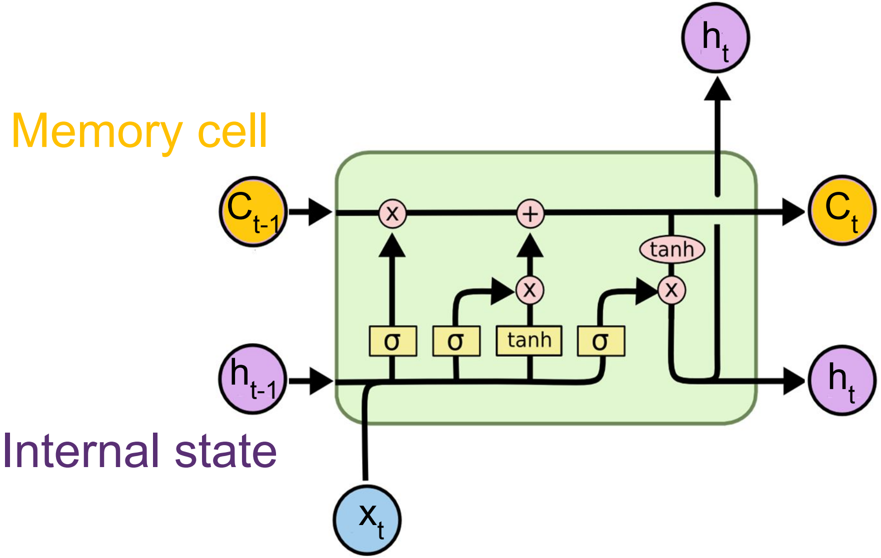
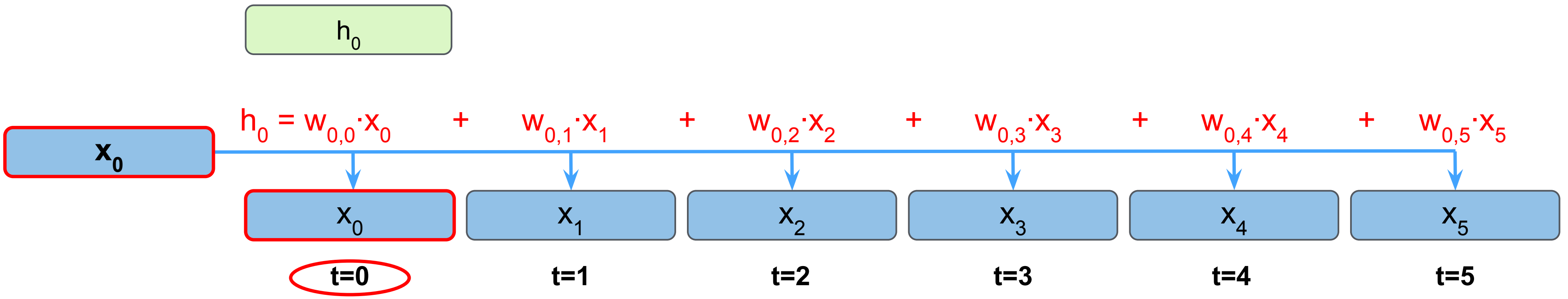


Image from Christopher Olah's blog

Even Longer Sequences: Self-Attention

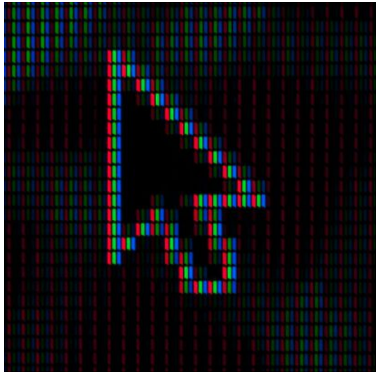
S	e	q	u	e	n	c	e		l	e	n	g	t	h

- Can be parallelized.
- Information can flow much better.



Is Self-Attention Perfect?

- It works well with sentences, paragraphs, ...
- What about:
  - Books? For example, **80000** elements (words).
  - Images? For example, **4096** elements (pixels) for a 64x64 image.
  - Sound? For example, **44100** elements (data points) for 1 second of CD quality audio.

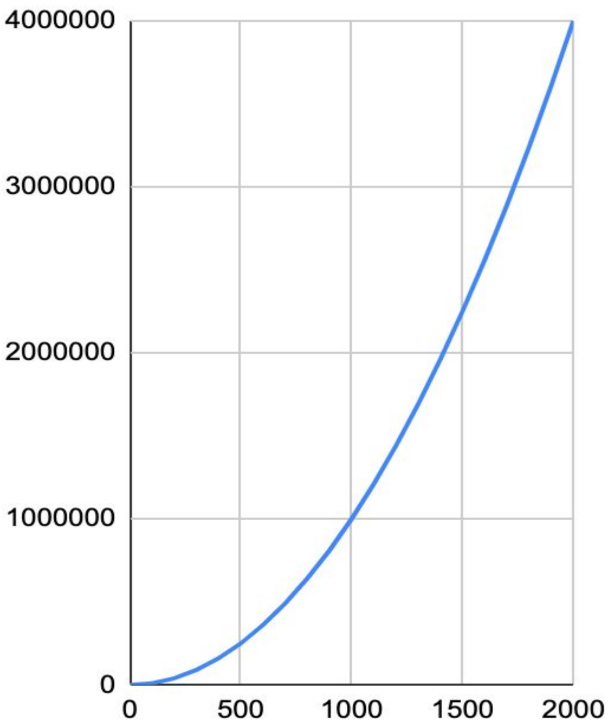


https://unsplash.com/photos/DgQf1dUKUTM, https://unsplash.com/photos/deb2EnbWPr8, https://unsplash.com/photos/OKLqGsCT8qs

Is Self-Attention Perfect? No..

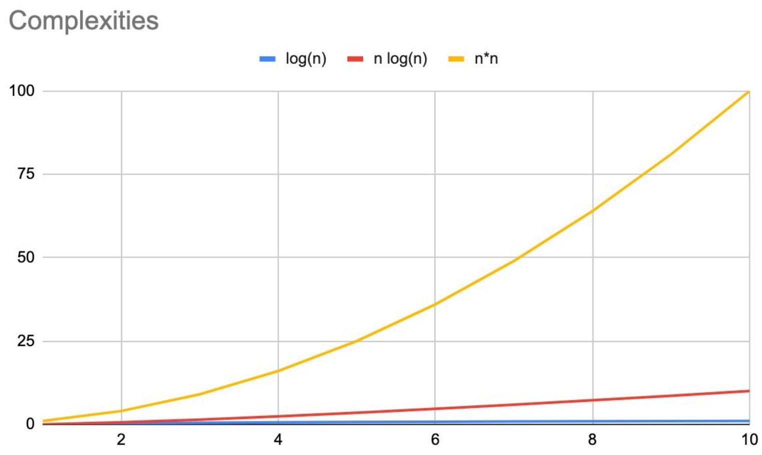
- Self-Attention scales quadratically with time/memory.
- Why? For **every** n element in the sequence, compare it with **every** n other element.
  So, $O(n*n) = O(n^2)$

- This is bad news..



Alert: What is Complexity? (1)

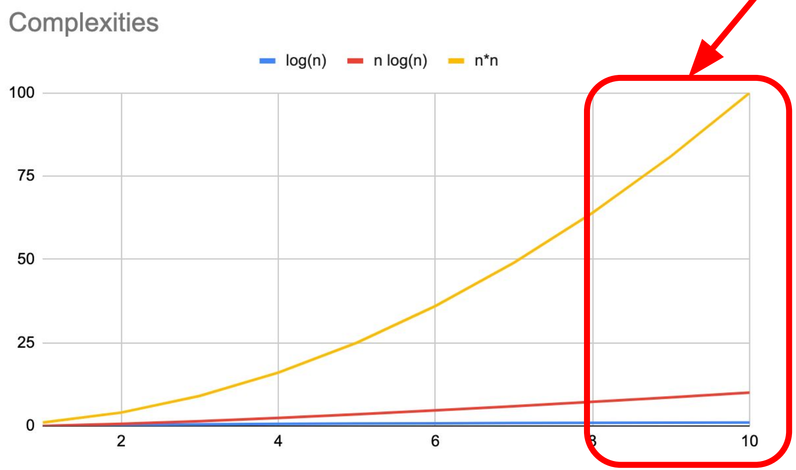
- **Asymptotic complexity** is the analysis of how an operation's cost scales with respect to some factors. For example:
  - The cost of finding an element in a sorted sequence of length n => O(log(n)) .
  - The cost of sorting a sequence with length n => O(n log(n)) .
  - The cost of applying Self-Attention on a sequence with length n => O(n*n) .



Mila

Alert: What is Complexity? (2)

- Why is it called asymptotic complexity?
- Because we are interested in what happens when the factor becomes big. (in the previous slide examples, when n becomes big)



Mila

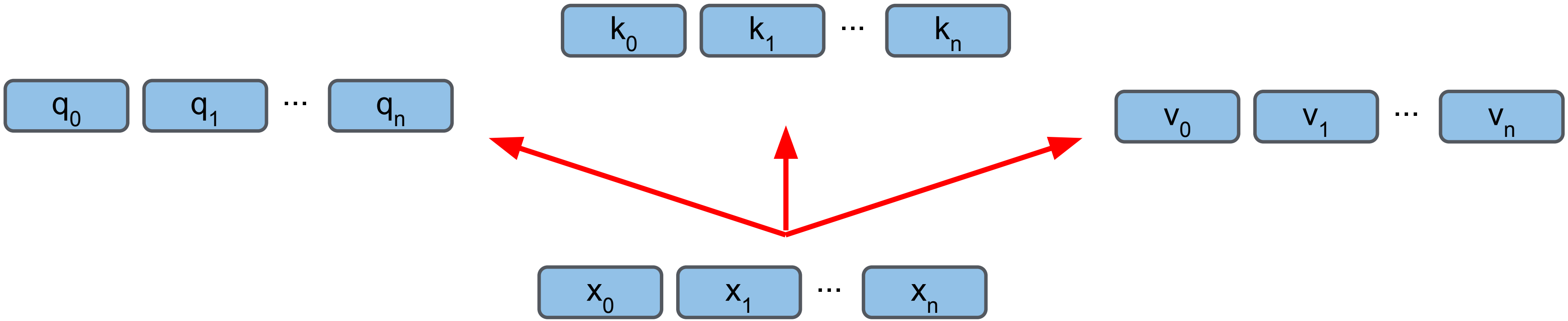
Self-Attention: Complexity (1)

- Self-Attention is implemented in vectorized form to be efficient.
- Let's confirm that the complexity - in vectorized form - is $O(n^2)$.

Self-Attention: Complexity (2)

- First, let's introduce some terminology: **queries** (Q), **keys** (K), **values** (V).

- In the videos, we saw that Self-Attention works with a single sequence X.
- That is true. But technically (to give more flexibility to the models), we generate three different sequences from X: Q, K, and V.



Mila

Self-Attention: Complexity (3)

- Self-Attention, takes every element (from the **queries**) and compare them with every other element (the **keys**).
- This comparison provides some weights. We multiply these weights by the **values** to get the final result.

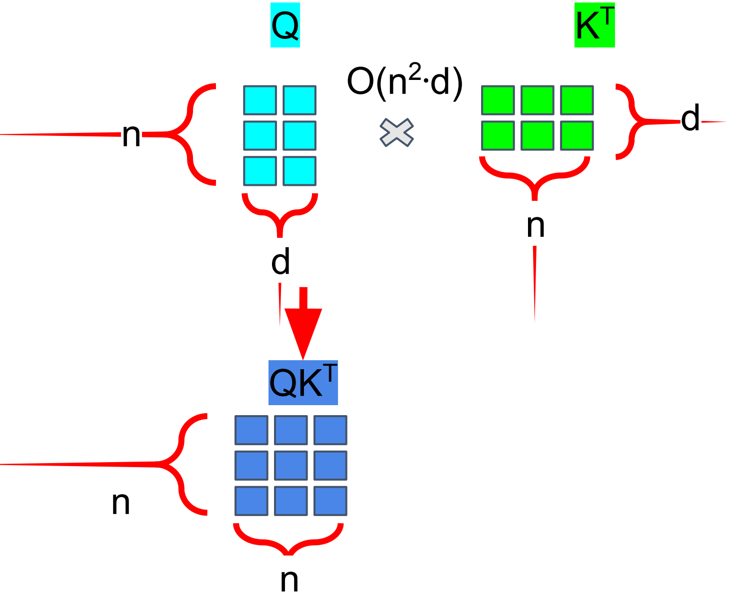
$$\text{Self-Attention}(Q, K, V) = (\frac{Softmax(QK^T)}{\sqrt{d}})V$$

We will ignore both the Softmax and the scaling factor d in the following slides.

Mila

Self-Attention: Complexity (4)

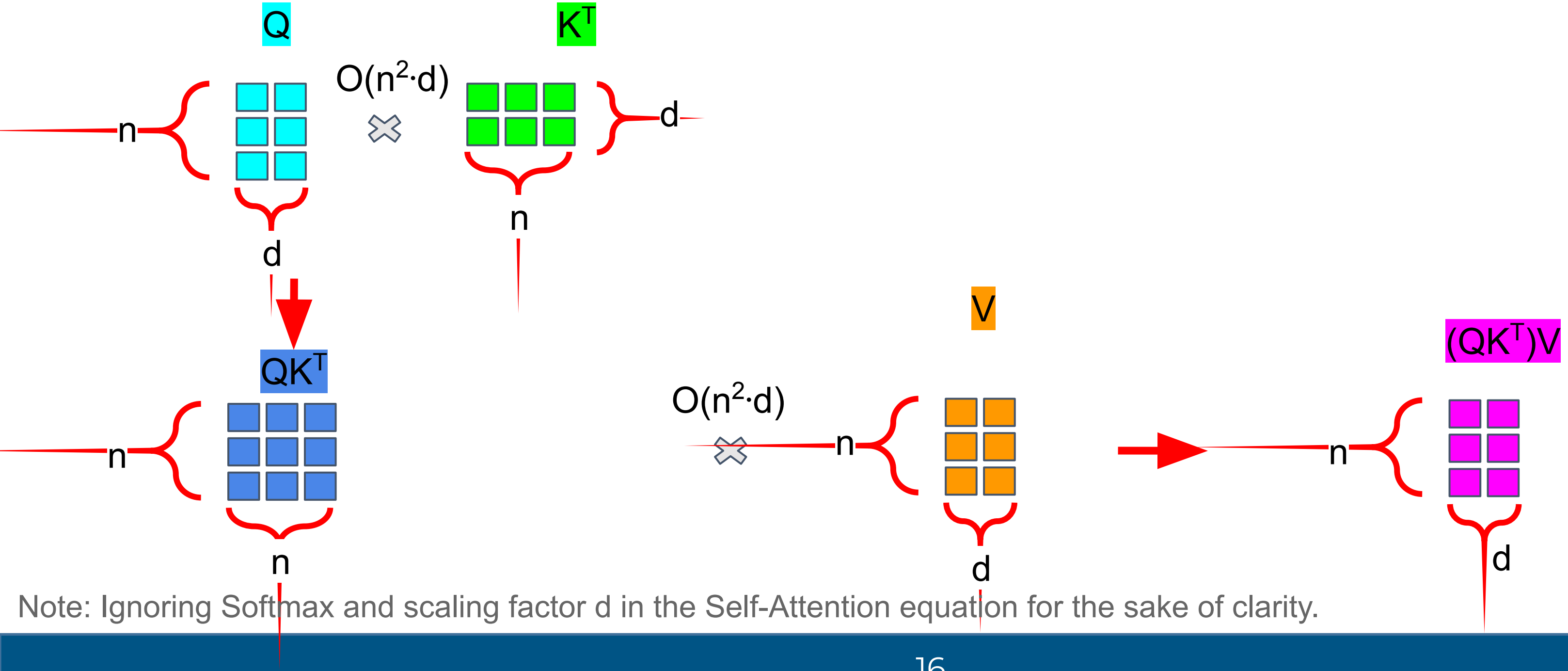
$$\text{Self-Attention}(Q, K, V) = \left(\frac{Softmax(QK^T)}{\sqrt{d}}\right)V$$



Note: Ignoring Softmax and scaling factor d in the Self-Attention equation for the sake of clarity.

Self-Attention: Complexity (5)

$$\text{Self-Attention}(Q, K, V) = \left(\frac{Softmax(QK^T)}{\sqrt{d}}\right)V$$



Note: Ignoring Softmax and scaling factor d in the Self-Attention equation for the sake of clarity.

Researchers to the Rescue!

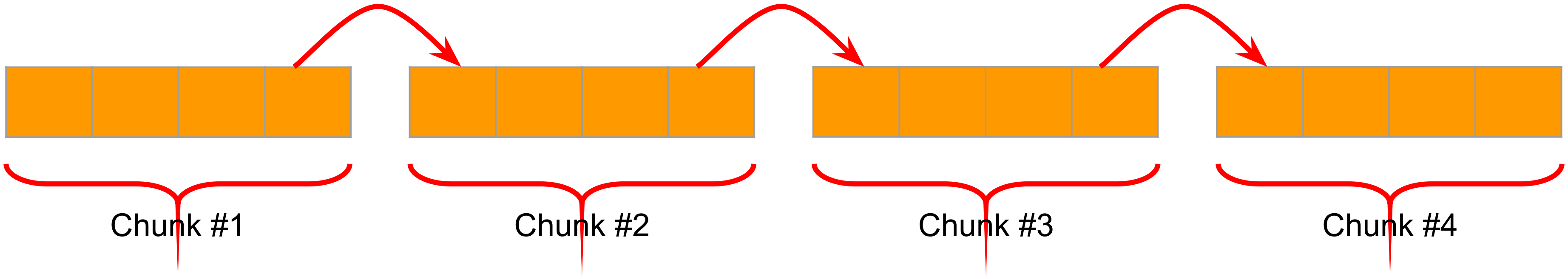
- So, Self-Attention has quadratic complexity, and that is bad news.

- We will now investigate some interesting directions to improve Self-Attention complexity:
    - Let's use memory!
    - Let's make Self-Attention computationally less costly!
    - Let's limit Self-Attention to some specific patterns!

Re-Adding Memory to Transformers

- Transformer-XL aims at handling long sequences.

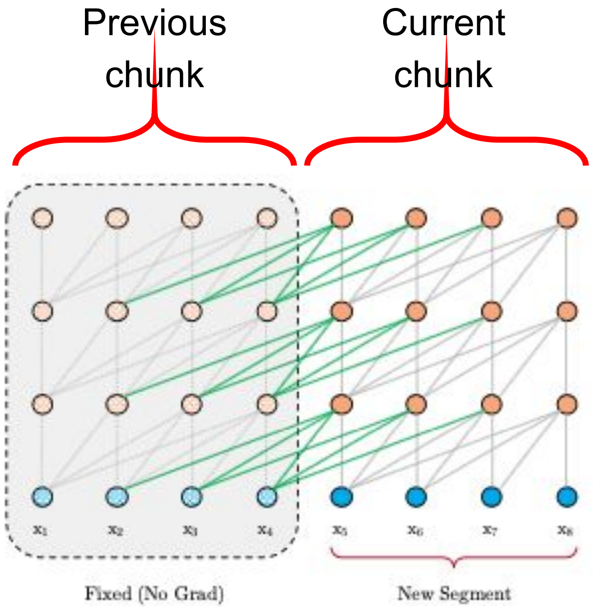
- The idea is simple:
  - Split the sequences into chunks.
  - Apply Self-Attention **one chunk at a time**, but considering also the previous chunk.



Dai et al. "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context."

Transformer-XL (1)

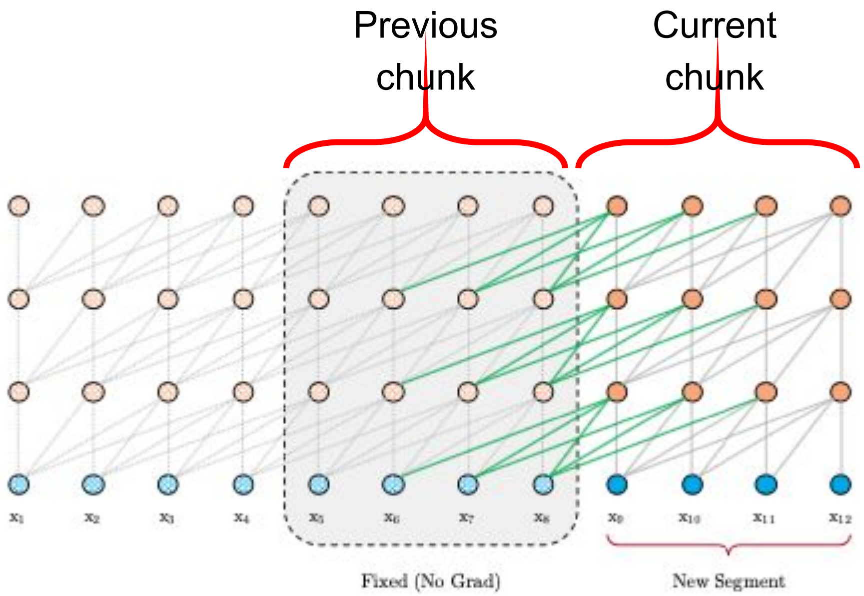
- Self-Attention has access to the current chunk...
- ...but also to the previous chunk.
- This creates a "**memory** effect".



Dai et al. "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context."

Transformer-XL (2)

- The same process is repeated for every chunk.
- Note how **only the previous chunk** is being accessed by Self-Attention (and not the full past).
- This is what lower the computational requirements.



Dai et al. "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context."

Transformer-XL: Performances

- Good results on Language Modeling.

Model	#Params	Validation PPL	Test PPL
Grave et al. (2016b) – LSTM	-	-	48.7
Bai et al. (2018) – TCN	-	-	45.2
Dauphin et al. (2016) – GCNN-8	-	-	44.9
Grave et al. (2016b) – LSTM + Neural cache	-	-	40.8
Dauphin et al. (2016) – GCNN-14	-	-	37.2
Merity et al. (2018) – 4-layer QRNN	151M	32.0	33.0
Rae et al. (2018) – LSTM + Hebbian + Cache	-	29.7	29.9
Ours – Transformer-XL Standard	151M	**23.1**	**24.0**
Baevski & Auli (2018) – adaptive input◇	247M	19.8	20.5
Ours – Transformer-XL Large	257M	**17.7**	**18.3**

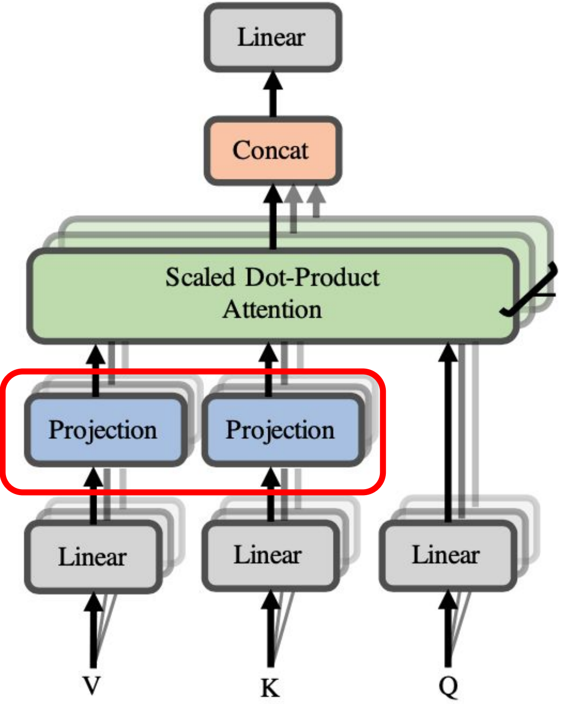
- Significant speed-up on long sequences.

Attn Len	How much Al-Rfou et al. (2018) is slower than ours
3,800	1,874x
2,800	1,409x
1,800	773x
800	363x

Dai et al. "Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context."

Can We Make Self-Attention Less Costly?

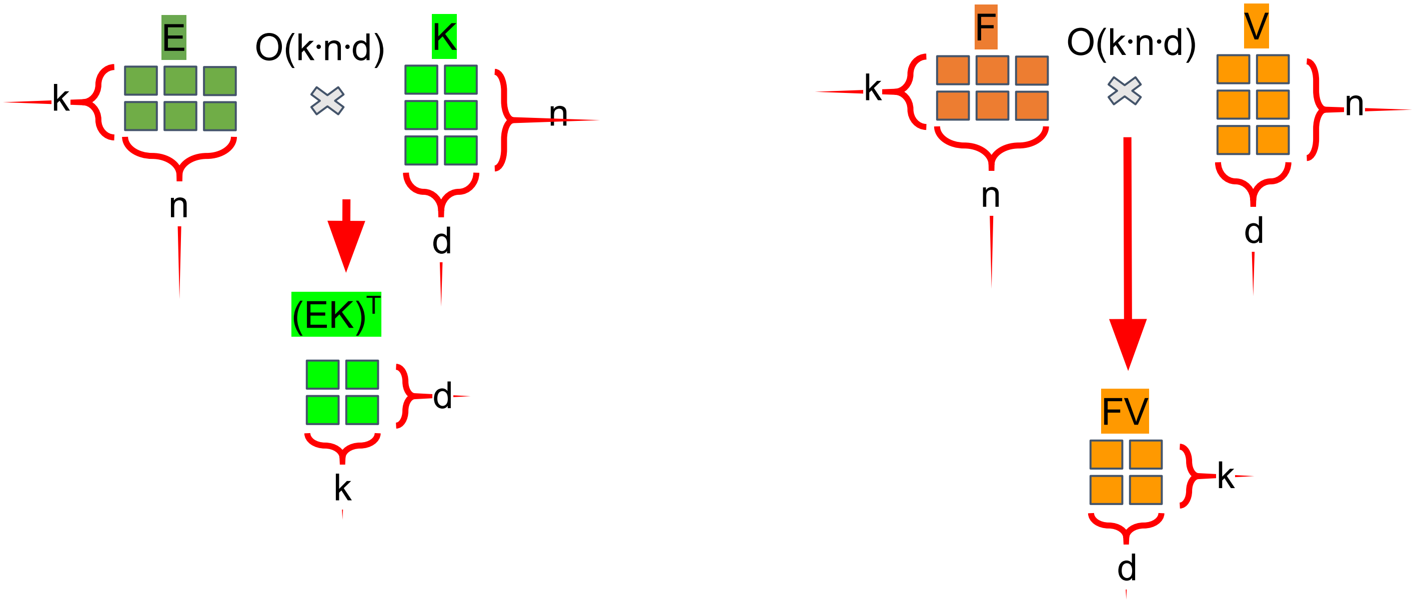
- We can use low-rank approximations of the Self-Attention matrix to lower the complexity.
- For example, the Linformer adds projections (matrix multiplications) to lower the matrix dimensions.



Wang et al. "Linformer: Self-Attention with Linear Complexity."

Linformer: Complexity (1)

- Two additional projections are added: E and F.

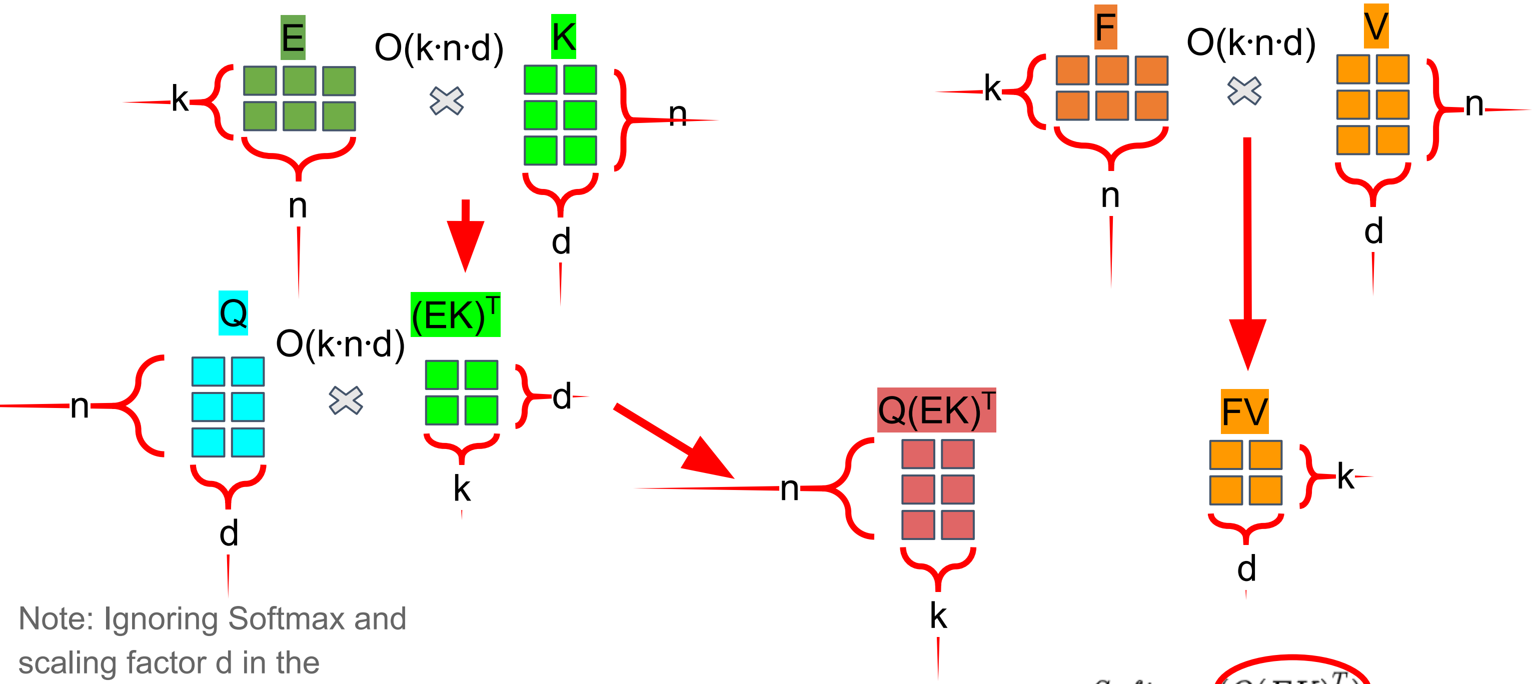


Note: Ignoring Softmax and scaling factor d in the Self-Attention equation for the sake of clarity.

$$\text{Self-Attention}(Q, K, V, E, F) = \left(\frac{Softmax(Q(EK)^T)}{\sqrt{d}}\right)FV$$

Linformer: Complexity (2)

- Two additional projections are added: E and F.



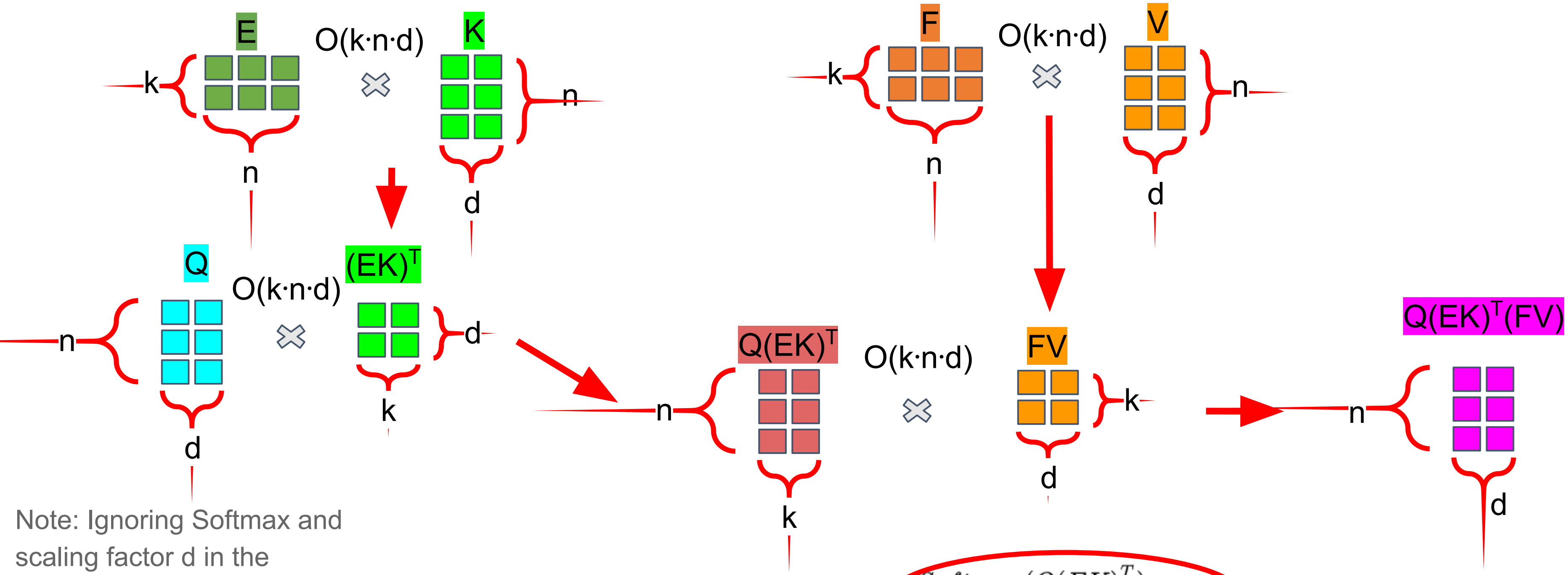
Note: Ignoring Softmax and scaling factor d in the Self-Attention equation for the sake of clarity.

$$\text{Self-Attention}(Q, K, V, E, F) = \left(\frac{Softmax(Q(EK)^T)}{\sqrt{d}}\right)FV$$

Mila

Linformer: Complexity (3)

- Two additional projections are added: E and F.



Note: Ignoring Softmax and scaling factor d in the Self-Attention equation for the sake of clarity.

$$\text{Self-Attention}(Q, K, V, E, F) = \left(\frac{Softmax(Q(EK)^T)}{\sqrt{d}}\right)FV$$

Mila

Complexity Comparison

- The complexity of the original Self-Attention is $O(n^2 \cdot d)$.
- The complexity of the Linformer is $O(k \cdot n \cdot d)$.

- Here, both d and k are fixed. So, we are interested only in n.
- As a consequence, original Self-Attention is $O(n^2)$, while Linformer is $O(n)$.

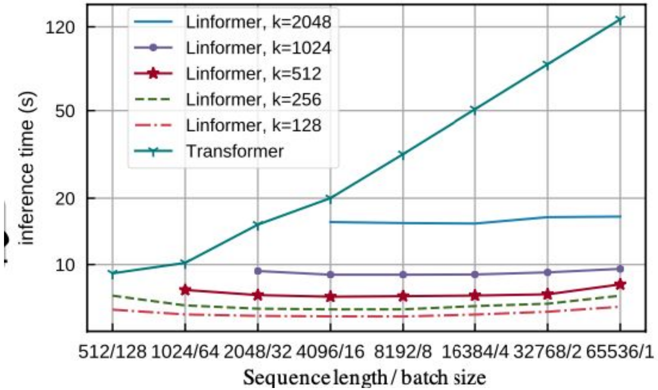
- These matrix dimension reductions may come at the cost of performances.
  Is this the case in the results?

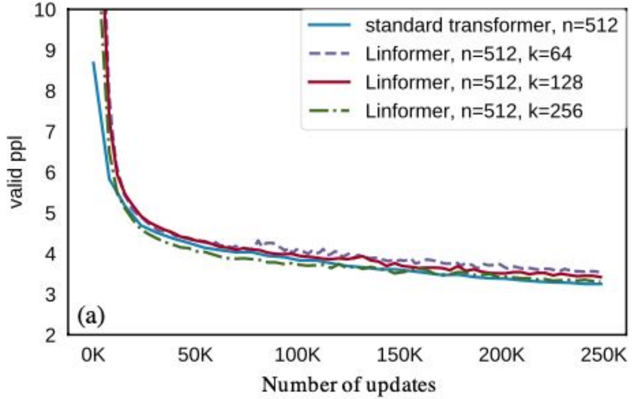
Note: Ignoring the Softmax in the Self-Attention equation for the sake of clarity.

Linformer: Results

- Results show faster computation...
- ... and no noticeable drop on results.

  (results are on Language Modeling)





Wang et al. "Linformer: Self-Attention with Linear Complexity."

Linformer: Results

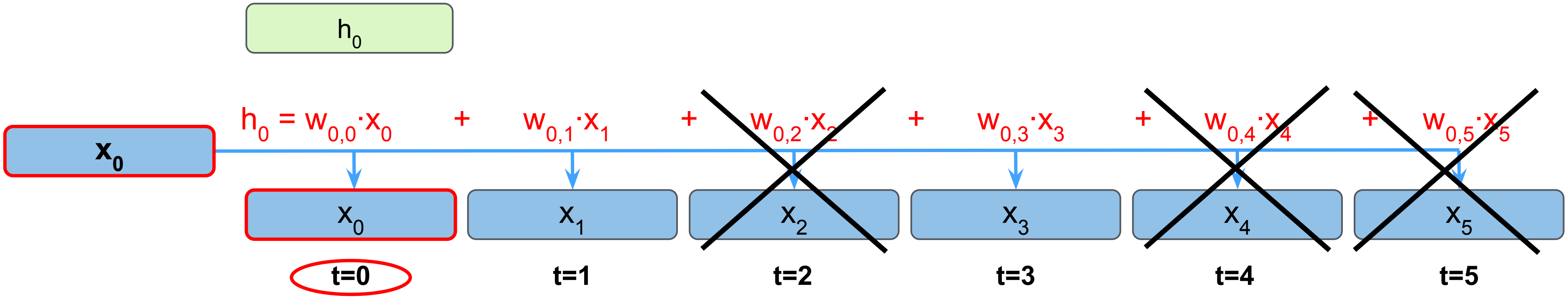
- What about results on classification tasks?
- Linformer matches (sometime improves) state-of-the-art results.

n	Model	SST-2	IMDB	QNLI	QQP	Average
512	Liu et al. (2019), RoBERTa-base	93.1	94.1	90.9	**90.9**	92.25
	Linformer, 128	92.4	94.0	90.4	90.2	91.75
	Linformer, 128, shared kv	**93.4**	93.4	90.3	90.3	91.85
	Linformer, 128, shared kv, layer	93.2	93.8	90.1	90.2	91.83
	Linformer, 256	93.2	94.0	90.6	90.5	92.08
	Linformer, 256, shared kv	93.3	93.6	90.6	90.6	92.03
	Linformer, 256, shared kv, layer	93.1	94.1	**91.2**	90.8	**92.30**
512	Devlin et al. (2019), BERT-base	92.7	93.5	91.8	89.6	91.90
	Sanh et al. (2019), Distilled BERT	91.3	92.8	89.2	88.5	90.45
1024	Linformer, 256	93.0	93.8	90.4	90.4	91.90
	Linformer, 256, shared kv	93.0	93.6	90.3	90.4	91.83
	Linformer, 256, shared kv, layer	93.2	**94.2**	90.8	90.5	92.18

Wang et al. "Linformer: Self-Attention with Linear Complexity."

Can We Limit the Self-Attention Span? (1)

- We can save computation by allowing Self-Attention **to only look at some elements**.
- Which elements? We need to find patterns that make sense with the data we are dealing with.



Can We Limit the Self-Attention Span? (2)

- The Sparse Transformer does that.
- In the papers, authors experiment with both images and text.

Child et al. "Generating Long Sequences with Sparse Transformers."

Alert: Images as Sequences?

- An image can be easily flattened as a sequence:

1	2	3
4	5	6
7	8	9

1	2	3	4	5	6	7	8	9

- Do we lose information by doing so (given we lose the 2d structure)?
- In theory yes, but in practice we hope that Self-Attention will be able to "recover" the underlying 2d structure..

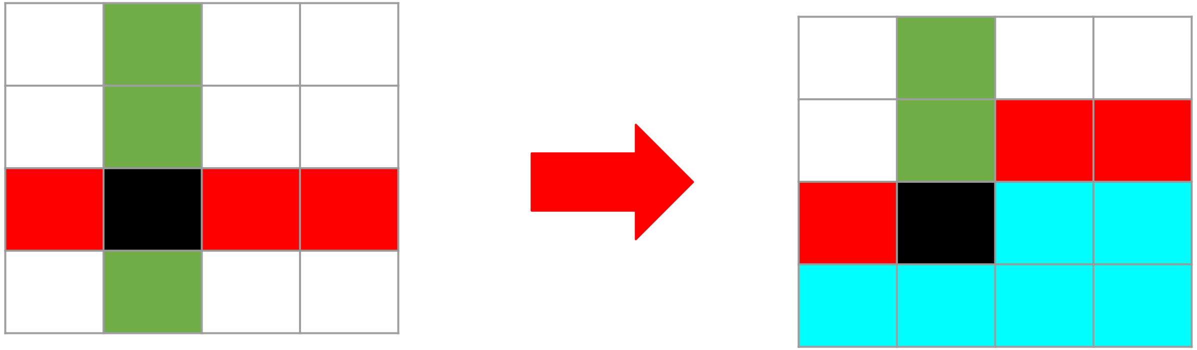
Patterns For Images (1)

- On sequences derived by flattening images, the Sparse Transformer looks at the same row/column only. This is called **strided** pattern.

Child et al. "Generating Long Sequences with Sparse Transformers."

Patterns For Images (2)

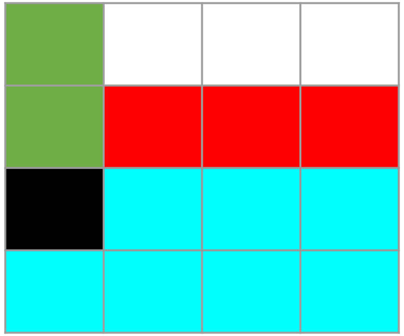
- On sequences derived by flattening images, the Sparse Transformer looks at the same row/column only. This is called **strided** pattern.
- Note: here we consider Self-Attention that does not look at future elements (this is called causal attention).



Child et al. "Generating Long Sequences with Sparse Transformers."

Patterns For Images: Example (1)

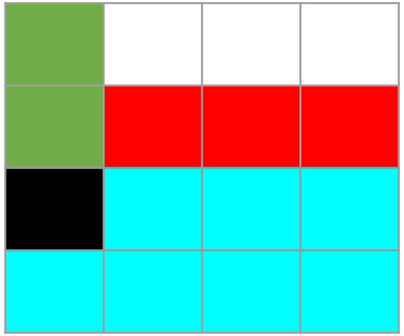
- On sequences derived by flattening images, the Sparse Transformer looks at the same row/column only. This is called **strided** pattern.
- Note: here we consider Self-Attention that does not look at future elements (this is called causal attention).

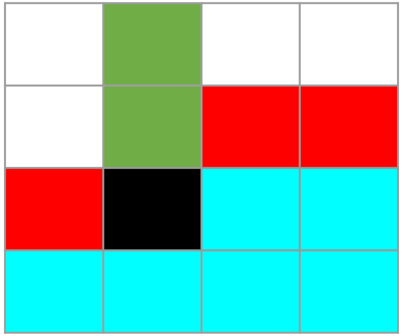


Child et al. "Generating Long Sequences with Sparse Transformers."

Patterns For Images: Example (2)

- On sequences derived by flattening images, the Sparse Transformer looks at the same row/column only. This is called **strided** pattern.
- Note: here we consider Self-Attention that does not look at future elements (this is called causal attention).

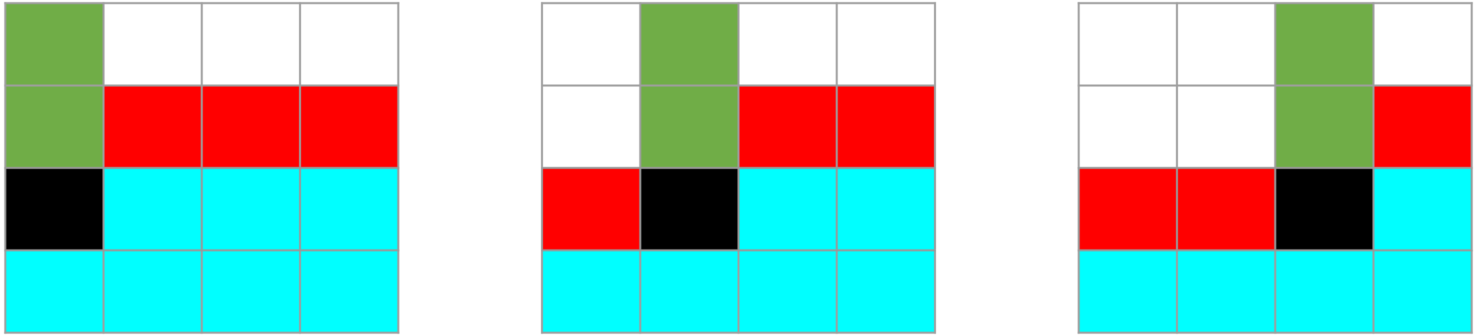




Child et al. "Generating Long Sequences with Sparse Transformers."

Patterns For Images: Example (3)

- On sequences derived by flattening images, the Sparse Transformer looks at the same row/column only. This is called **strided** pattern.
- Note: here we consider Self-Attention that does not look at future elements (this is called causal attention).



Child et al. "Generating Long Sequences with Sparse Transformers."

Patterns For Text (1)

- Does it also make sense with text?
- Text does not have a 2-dimensional structure, so, strided pattern is not effective.

- The authors use instead a **fixed** pattern, which is: look at neighbours plus at some fixed positions.

- Why fixed positions?
  - They allow the model to look further away.
  - They provide stability to the model.

Child et al. "Generating Long Sequences with Sparse Transformers."

Patterns For Text (2)

- The authors use a **fixed** pattern, which is: look at neighbours plus at some fixed positions.
- Note: here we consider Self-Attention that does not look at future elements (this is called causal attention).

Child et al. "Generating Long Sequences with Sparse Transformers."

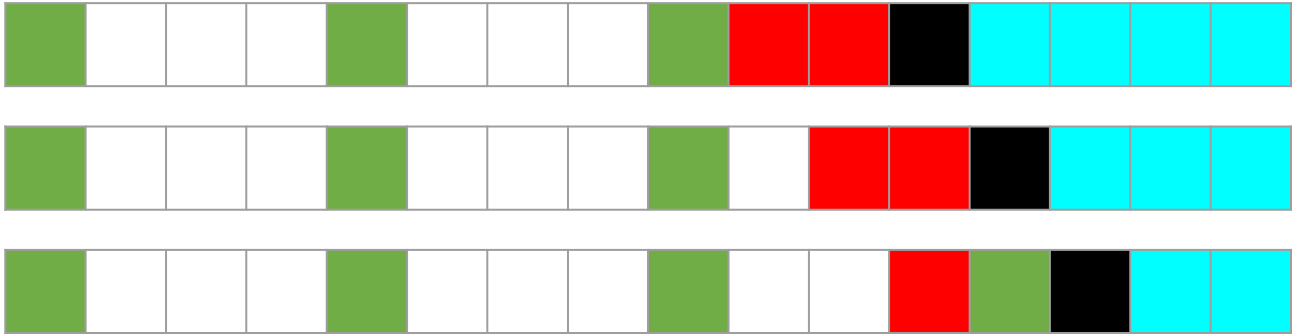
Patterns For Text (3)

- The authors use a **fixed** pattern, which is: look at neighbours plus at some fixed positions.
- Note: here we consider Self-Attention that does not look at future elements (this is called causal attention).

Child et al. "Generating Long Sequences with Sparse Transformers."

Patterns For Text (4)

- The authors use a **fixed** pattern, which is: look at neighbours plus at some fixed positions.
- Note: here we consider Self-Attention that does not look at future elements (this is called causal attention).



Child et al. "Generating Long Sequences with Sparse Transformers."

Sparse Transformer: Results

- The Sparse Transformer provides good results on Language Modeling.

Enwik8	Bits per byte
Deeper Self-Attention (Al-Rfou et al., 2018)	1.06
Transformer-XL 88M (Dai et al., 2018)	1.03
Transformer-XL 277M (Dai et al., 2018)	**0.99**
Sparse Transformer 95M (fixed)	**0.99**

Model	Bits per byte
CIFAR-10	
PixelCNN (Oord et al., 2016)	3.03
PixelCNN++ (Salimans et al., 2017)	2.92
Image Transformer (Parmar et al., 2018)	2.90
PixelSNAIL (Chen et al., 2017)	2.85
Sparse Transformer 59M (strided)	**2.80**

- With a significant speed-up on computation.

Model	Bits per byte	Time/Iter
Enwik8 (12,288 context)		
Dense Attention	1.00	1.31
Sparse Transformer (Fixed)	**0.99**	0.55
Sparse Transformer (Strided)	1.13	0.35

Child et al. "Generating Long Sequences with Sparse Transformers."

Final Thoughts on Efficient Transformers

- These Transformer architectures that work with long sequences are called **Efficient Transformers**.
    - Or, sometimes, **Long Transformers**.

- There are many papers and interesting ideas in this field.
- For a very interesting survey, see Tay et al. "Efficient Transformers: A Survey".

- Also, if you work with images/sound, Efficient Transformers will probably be very important there (in the near future).

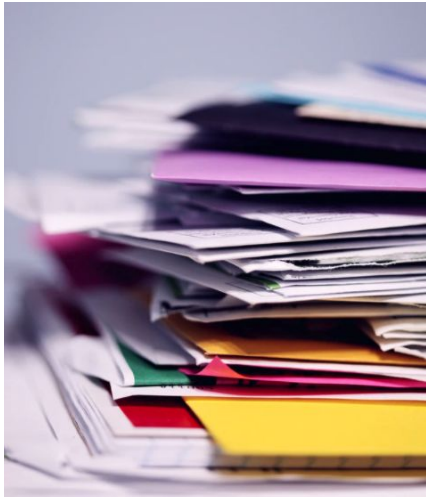
Plan

- Life after Transformers
- **Life after BERT**
- Tools for NLP

What happened after BERT?

How did we arrive to BERT? (1)

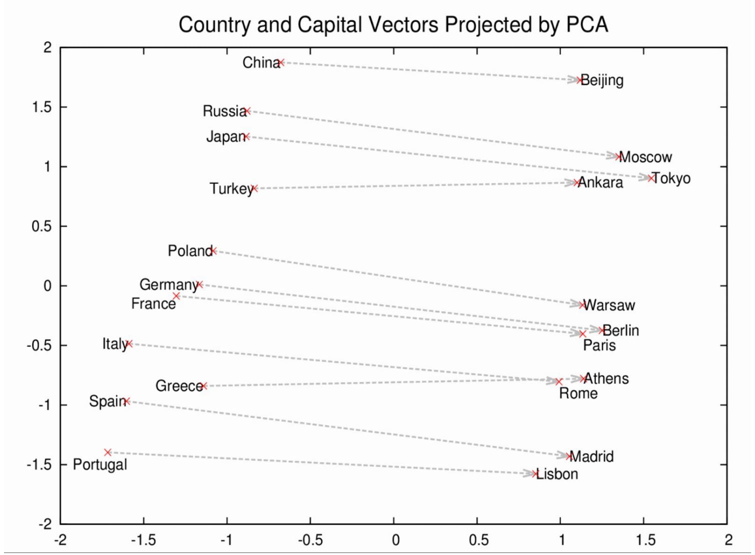
- We started by introducing pre-training (a transfer learning technique) and fine-tuning:
- First collect general syntactic and semantic knowledge with a self-supervised task on a lot of data (**pre-training**)...
- ...then train on the task of interest (**fine-tuning**).



https://unsplash.com/photos/jedKD4yaTvk, https://unsplash.com/photos/tn57JI3CewI

How did we arrive to BERT? (2)

- Our pre-training journey started with Word2Vec.
- Word2Vec created distributed representations for tokens trying to capture their syntactic and semantic properties.

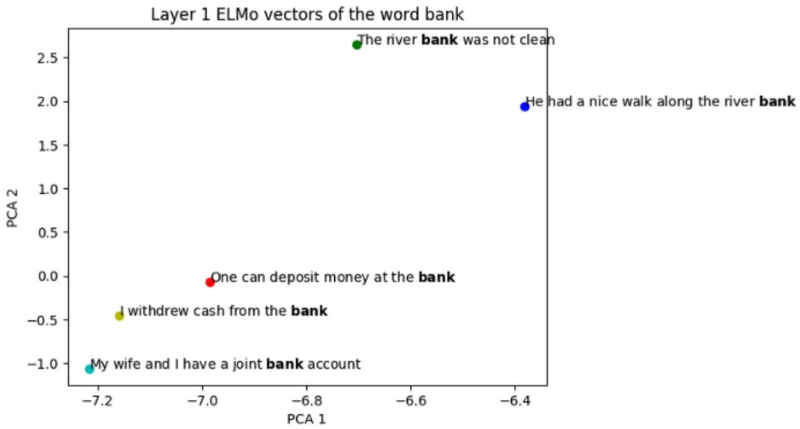


Mikolov et al. "Efficient estimation of word representations in vector space."

Mila

How did we arrive to BERT? (2)

- We improved on that by considering the **context** (when creating word representations). This was done in ELMo.

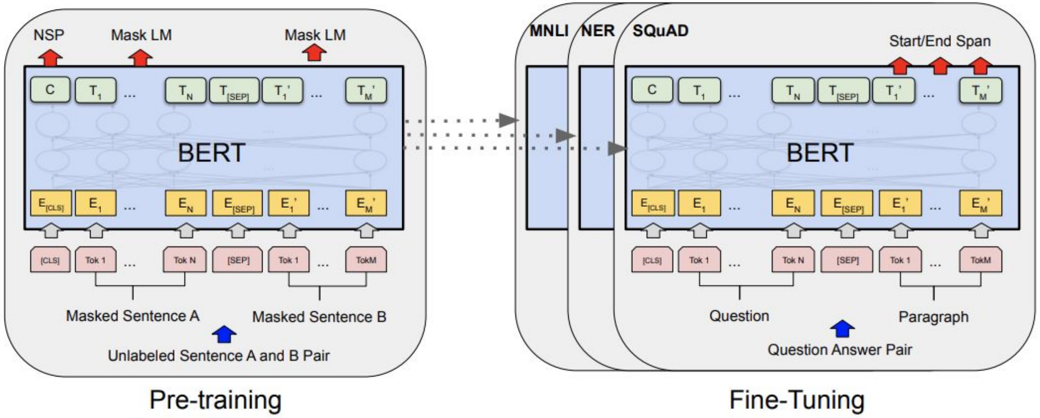


Peters et al. "Deep contextualized word representations."
https://unsplash.com/photos/2_K82gx9Uk8, https://unsplash.com/photos/bP6-6xXb-oQ

Mila

How did we arrive to BERT? (3)

- Then, BERT improved over ELMo by:
  - Using the Transformer architecture.
  - Using Masked-Language-Modeling task for pre-training (instead of Language Modeling).



Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding."

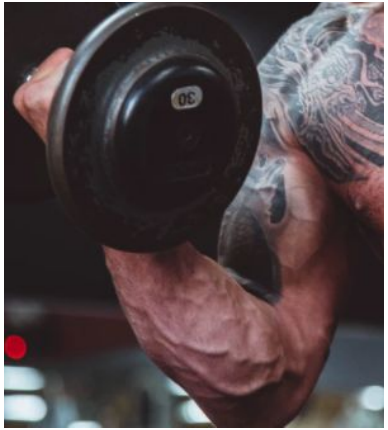
What happened after BERT?

- Many papers tried to improve.

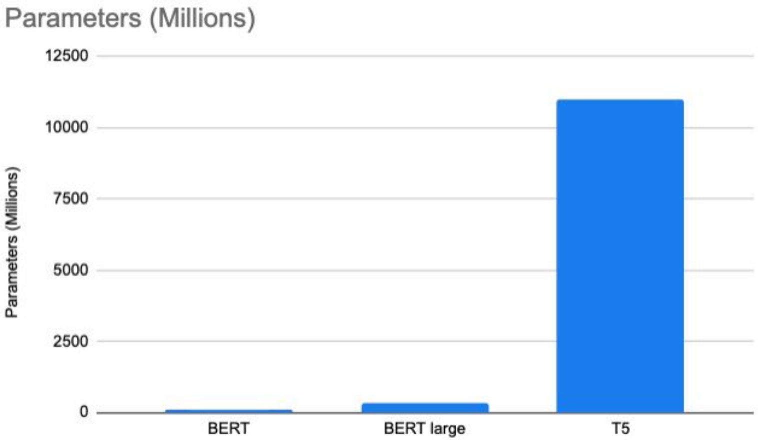
- Important directions:
  - We need bigger models!
  - We need more principled (maybe even smaller) models!
  - We do not want to fine-tune anymore!

Bigger Models

- Using bigger models means more computational power.



	Model: Parameters (Millions)
BERT base	110
BERT large	340
T5	11000

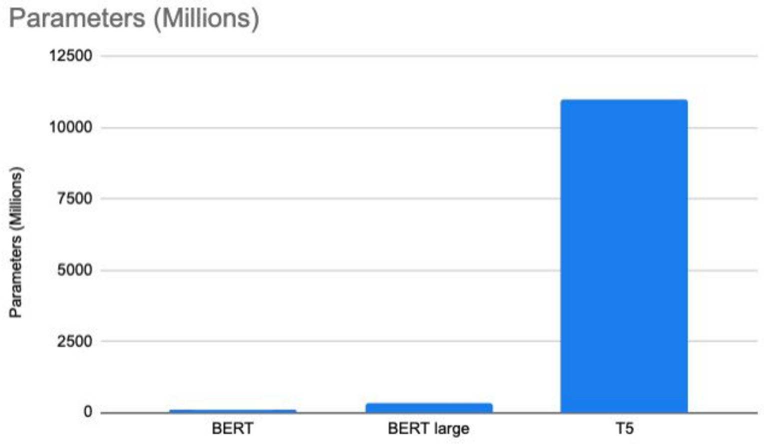


https://unsplash.com/photos/JNeYWQncbj8

Bigger Models

- Bigger models can also mean:
    - More pre-training data.
    - Different (or more) objective functions.
    - Variations in the training process to better exploit the added computational power.

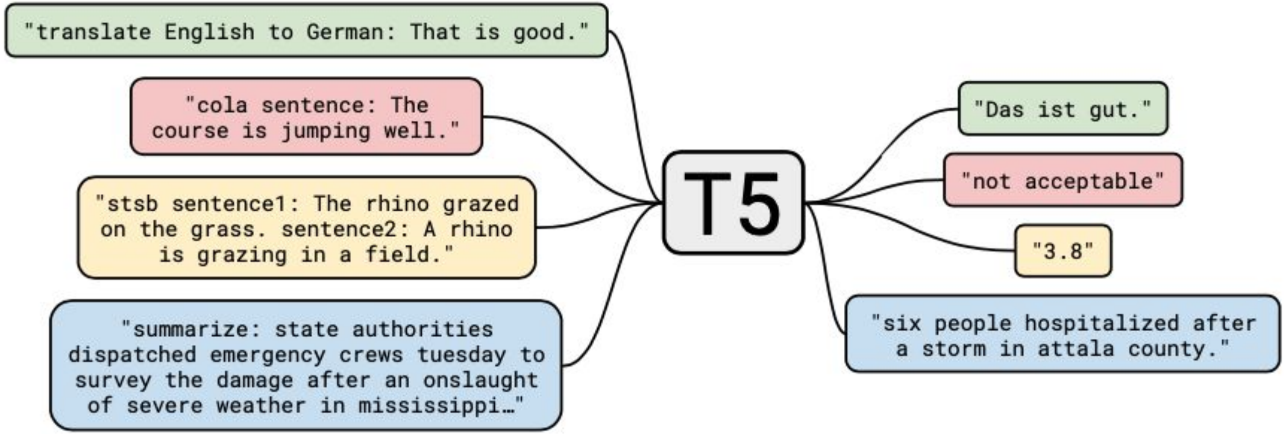
	Model: Parameters (Millions)
BERT base	110
BERT large	340
T5	11000



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

Text-to-Text Transfer Transformer (T5)

- To better exploit the big computational power, the **same** model is (elegantly) trained on more tasks.



- They introduce the "Colossal Clean Crawled Corpus" (C4), a dataset consisting of hundreds of gigabytes.

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

T5: (Some) Results

- T5 provides better results on some tasks...
- ...but not on all tasks.
  - Note though that several SOTA are ensemble of models (not just a single model, like T5).

Model	GLUE Average	CoLA Matthew's	SST-2 Accuracy	MRPC F1	MRPC Accuracy	STS-B Pearson	STS-B Spearman
Previous best	89.4[a]	69.2[b]	**97.1**[a]	**93.6**[b]	**91.5**[b]	**92.7**[b]	**92.3**[b]
T5-11B	**89.7**	**70.8**	**97.1**	91.9	89.2	92.5	92.1

Model	QQP F1	QQP Accuracy	MNLI-m Accuracy	MNLI-mm Accuracy	QNLI Accuracy	RTE Accuracy	WNLI Accuracy
Previous best	**74.8**[c]	**90.7**[b]	91.3[a]	91.0[a]	**99.2**[a]	89.2[a]	91.8[a]
T5-11B	74.6	90.4	**92.0**	**91.7**	96.7	**92.5**	**93.2**

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

More Principled Models

- Some papers claim instead that big models are over-parameterized.
- Which means that we can achieve the same (or better) results with smaller/mode efficient models.

https://www.freepik.com/premium-vector/less-is-more-neon-signs-style-text_8249155.htm

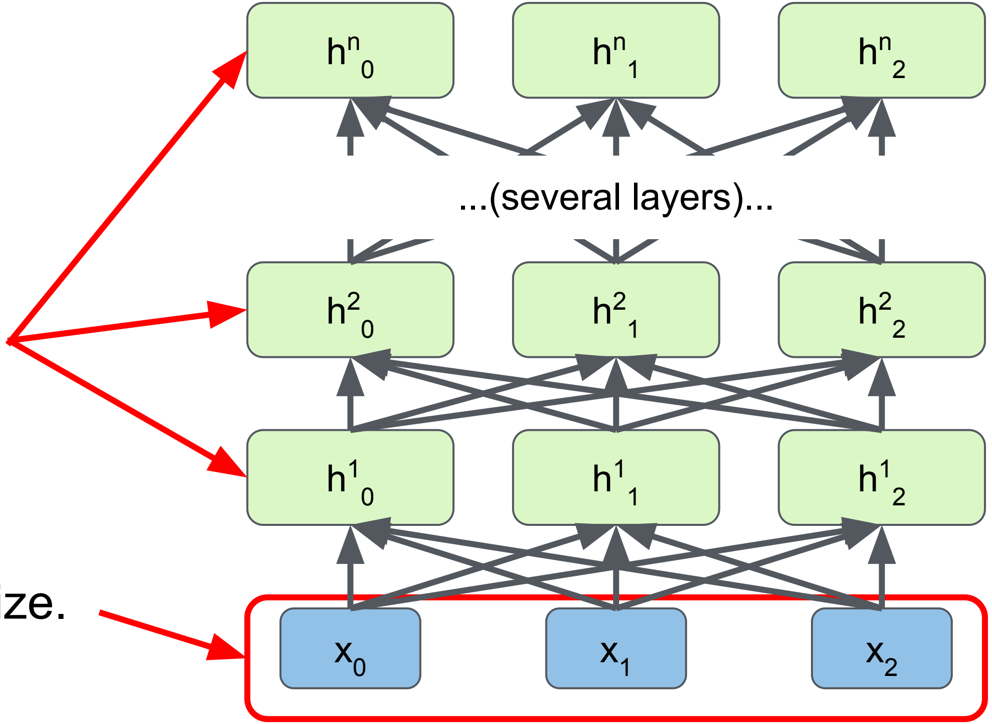
A Lite BERT - ALBERT (1)

- ALBERT aims at having a smaller and more efficient model by:
    - Cross-layer parameter sharing.
    - Reducing the input embedding size (but **not** the hidden state size).

Lan et al. "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations."

A Lite BERT - ALBERT (2)

- Cross-layer parameter sharing.
- Reducing the input embedding size.



Lan et al. "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations."

ALBERT: Results

- ALBERT models (except xxlarge) are smaller than any BERT models...
- ...and they provide very competitive (if not better) results.

Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.5/83.3	80.3/77.3	84.1	91.7	68.3	82.1	17.7x
	large	334M	92.4/85.8	83.9/80.8	85.8	92.2	73.8	85.1	3.8x
	xlarge	1270M	86.3/77.9	73.8/70.5	80.5	87.8	39.7	76.7	1.0
ALBERT	base	12M	89.3/82.1	79.1/76.1	81.9	89.4	63.5	80.1	21.1x
	large	18M	90.9/84.1	82.1/79.0	83.8	90.6	68.4	82.4	6.5x
	xlarge	59M	93.0/86.5	85.9/83.1	85.4	91.9	73.9	85.5	2.4x
	xxlarge	233M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	1.2x

Lan et al. "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations."

Do We Need Fine-Tuning?

- Is it possible that pre-training is all we need?
- If we can avoid fine-tuning, then one model is enough to rule all the NLP tasks!

https://pixabay.com/photos/ring-lord-of-the-rings-hobbit-4612457/

GPT-3 (1)

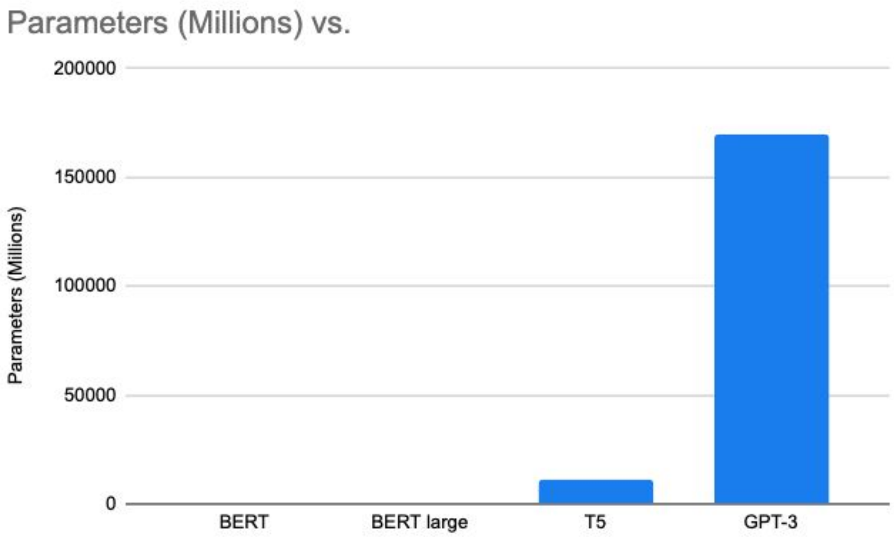
- GPT-3 abandons fine-tuning.
- It only does pre-training (using the Language Modeling task).

Brown et al. "Language Models are Few-Shot Learners."

GPT-3 (2)

- It is a huge model.
  (remember that T5 was a "big" model, a few slides ago..)

	Model: Parameters (Millions)
BERT base	110
BERT large	340
T5	11000
GPT-3	175000



GPT-3: Results

- The GPT-3 paper provides many results*:
  - Several state-of-the-art results.
  - Many good results.
  - Some less good results.

Setting	Winograd	Winogrande (XL)
Fine-tuned SOTA	**90.1**[a]	**84.6**[b]
GPT-3 Zero-Shot	88.3*	70.2
GPT-3 One-Shot	89.7*	73.2
GPT-3 Few-Shot	88.6*	77.7

Setting	LAMBADA (acc)	LAMBADA (ppl)	StoryCloze (acc)	HellaSwag (acc)
SOTA	68.0[a]	8.63[b]	**91.8**[c]	**85.6**[d]
GPT-3 Zero-Shot	**76.2**	**3.00**	83.2	78.9
GPT-3 One-Shot	**72.5**	**3.35**	84.7	78.1
GPT-3 Few-Shot	**86.4**	**1.92**	87.7	79.3

Setting	En→Fr	Fr→En	En→De	De→En	En→Ro	Ro→En
SOTA (Supervised)	**45.6**[a]	35.0 [b]	**41.2**[c]	40.2[d]	**38.5**[e]	**39.9**[e]
XLM [LC19]	33.4	33.3	26.4	34.3	33.3	31.8
MASS [STQ+19]	37.5	34.9	28.3	35.2	35.2	33.1
mBART [LGG+20]	-	-	29.8	34.0	35.0	30.5
GPT-3 Zero-Shot	25.2	21.2	24.6	27.2	14.1	19.9
GPT-3 One-Shot	28.3	33.7	26.2	30.4	20.6	38.6
GPT-3 Few-Shot	32.6	39.2	29.7	40.6	21.0	39.5

* Many more results can be found in the paper: Brown et al. "Language Models are Few-Shot Learners."

GPT-3: Why So Important? (1)

- As we saw, it is **not** fine-tuned.

- What does it mean?
The pre-trained model can be downloaded and used right away.

- Also, it seems to hint to the fact that just solving the pre-training objective (Language Modeling) is enough to achieve human intelligence on language...

https://unsplash.com/photos/tGBXiHcPKrM

GPT-3: Why So Important? (2)

- ...but there are (important) limitations:

- GPT-3 is not as good as humans in reasoning and text synthesis:
    - "GPT-3 [...] repeat themselves semantically at the document level, start to lose coherence over sufficiently long passages, contradict themselves, and occasionally contain non-sequitur sentences or paragraphs".
    - "GPT-3 seems to have special difficulty with “common sense physics” ".

- Language Modeling may not be enough to achieve intelligence:
    - "[GPT-3] may eventually run into (or could already be running into) the limits of the pretraining objective".

Brown et al. "Language Models are Few-Shot Learners."

GPT-3: Why So Important? (3)

- To summarize: GPT-3 opens the floor to interesting discussions.
- GPT-3 is a huge model. Just training such a big model is an outstanding achievement.
- GPT-3 provides some impressive results.
- Still, analysis of the failures may indicate that we need more complex approaches (instead of just training on Language Modeling) to reach human intelligence on language.

Final Thoughts on NLP and Pre-Training

- Pre-training in NLP is now extremely used.
- There are plenty of models, everyone with its own advantages/disadvantages.
- If you have an NLP task, most likely you want to start with a pre-trained model.
- If you are wondering where you can find those models, wait for the next slide..

Plan

- Life after Transformers
- Life after BERT
- **Tools for NLP**

Where should I look at to start using pre-trained Transformers models (like BERT)?

🤗 Transformers

Tools for NLP

- Is there a single tool that includes everything I need?
- Almost..

- For classification tasks (word/single sentence/two sentence-classification) and sequence-to-sequence tasks, Hugging Face is a mature and well organized project that implements many interesting models.

Transformers

https://github.com/huggingface/transformers

Why Hugging Face?

- Ready to download pre-trained models!
  - Already fine-tuned models are also available, if you want!
- Very easy to use!
- New models added frequently!
- Supports PyTorch and Tensorflow!
- Cool icon!

How hard is it? (1)

- Assuming you just want to do extractive Question Answering with an already pre-trained/fine-tuned model:

```
from transformers import pipeline

model_name = 'distilbert-base-cased-distilled-squad'
qa_model = pipeline('question-answering', model_name)
example = {
    'question': 'What do we do at Mila?',
    'context': 'I work at Mila, in Montreal, in the Machine Learning Applied Research Team.'
               'At Mila, among other things, we study Deep Learning. It is really interesting!'
}
print(qa_model(example)['answer'])
```

```
study Deep Learning.
```

- That's it!

Mila

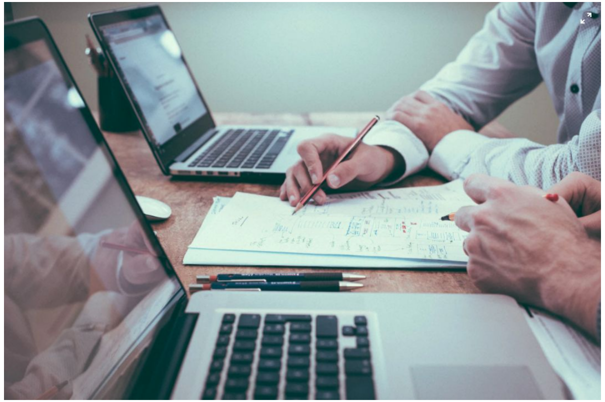
How hard is it? (2)

- Is that result cherry-picked?
  No (but note the example is not terribly hard).

`'deepset/roberta-base-squad2'`	→	`study Deep Learning.`
`'twmkn9/albert-base-v2-squad2'`	→	`study Deep Learning.`
`'mrm8488/bert-mini-finetuned-squadv2'`	→	`Deep Learning.`
`'mrm8488/bert-tiny-3-finetuned-squadv2'`	→	`Deep Learning.`

Is That A Realistic Use Case?

- Possibly, but in most cases you will want to fine-tune the model **on your data**.



- To this end, Hugging Face provides (among the others) a Trainer object that easily allow to fine-tune a pre-trained model.
- See https://huggingface.co/transformers/custom_datasets.html for more details.

https://unsplash.com/photos/5fNmWej4tAA

Which Models Are There?

- Big models for great computational capacity:
  t5-11b, bert-large-cased, gpt2, ...

- Small/efficient models for great performances in production:
  DistilBERT, ALBERT, ...

- Models for various languages:
  bert-base-japanese, bert-base-italian-cased, bert-base-finnish-cased-v1, ...

- Models for sequence-to-sequence NLP tasks:
  T5, BART, mBART, ...

https://huggingface.co/models

Final Thoughts on Hugging Face

- If you need an example about how to fine-tune some data with Hugging Face, see this week's tutorial.
- You will use a sequence-to-sequence Transformer model (T5) to perform machine translation.

- Hugging Face is a mature and efficient implementations for many Transformer models.
- If you have a NLP problem that can be shaped as classification or as sequence-to-sequence, Hugging Face should probably be your first option.

Quebec
Artificial
Intelligence
Institute

Mila

Questions?