VL Deep Learning for Natural Language Processing

18. Transformers

Prof. Dr. Ralf Krestel
AG Information Profiling and Retrieval

Learning Goals for this Chapter

- Understand transformers
- Know how BERT works and how to use it
- Understand current developments for word embeddings/language models

- Relevant chapters:
  - S9 (2021): https://www.youtube.com/watch?v=ptuGllU5SQQ
  - S10 (2021): https://www.youtube.com/watch?v=j9AcEI98C0o

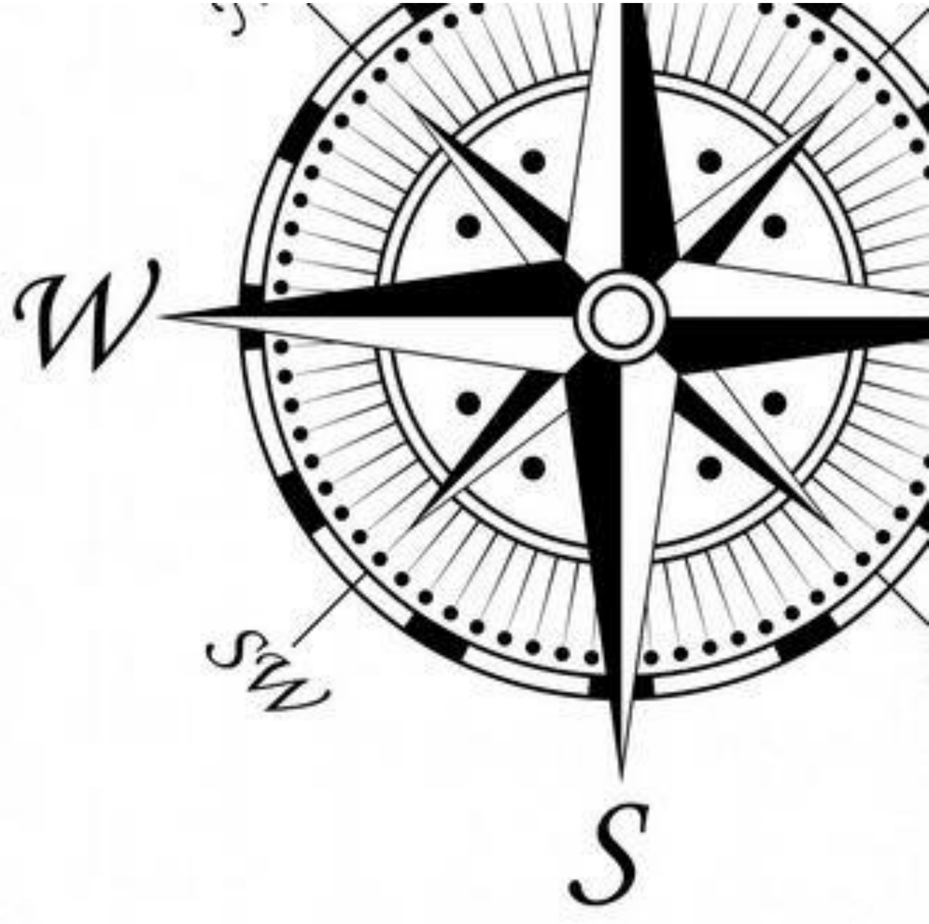
Let’s Scale it Up!

ULMfit	GPT	BERT	GPT-2	GPT-3
Jan 2018	June 2018	Oct 2018	Feb 2019	Juni 2020
Training: 1 GPU day	Training 240 GPU days	Training 256 TPU days ~320–560 GPU days	Training ~2048 TPU v3 days	Training 355 years on a Tesla V100 GPU
fast.ai	OpenAI	Google AI	OpenAI	OpenAI

Transformer Models

Topics Today

1. **Transformer**
2. BERT
3. Current Developments



Motivation

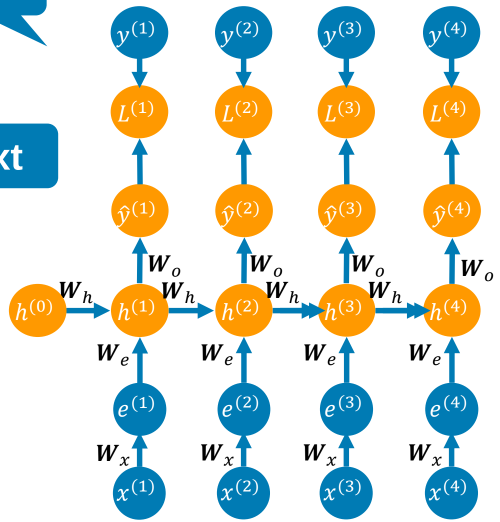
- We want to build large models,
  - Because they yield better results
- RNNs are sequencial models,
  - Thus, not parallelizable
- One-directional LM and BiLMs capture context only partly
- RNNs (LSTMs and also GRUs) need attention mechanism for long-range dependencies
- Attention allows access to all hidden states
- Why not discard RNNs completely and only use attention?

Small

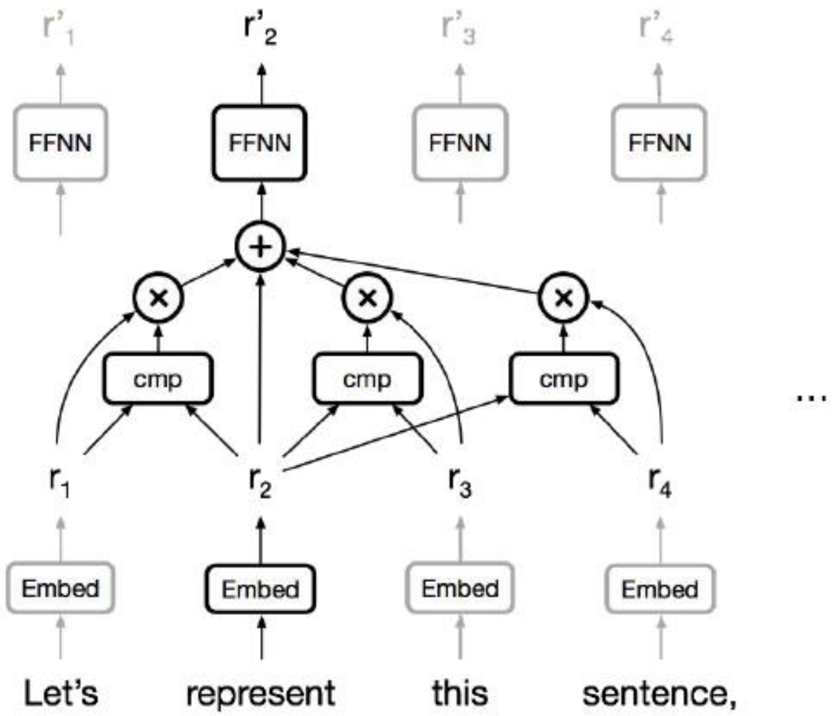
Slow

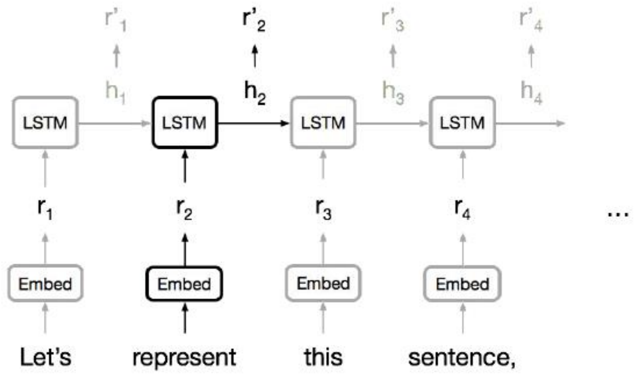
RNN

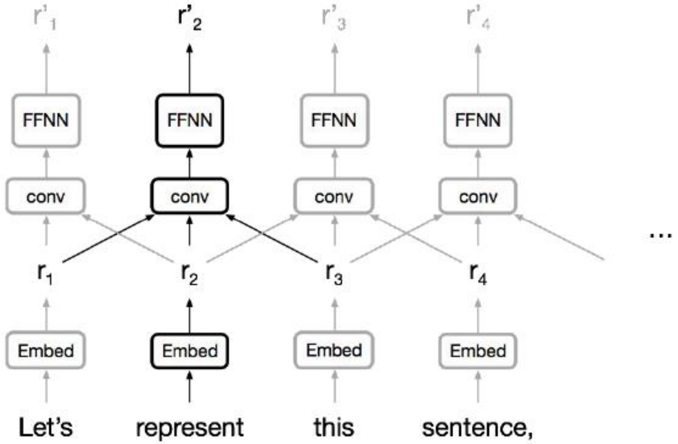
Limited context



Self-Attention

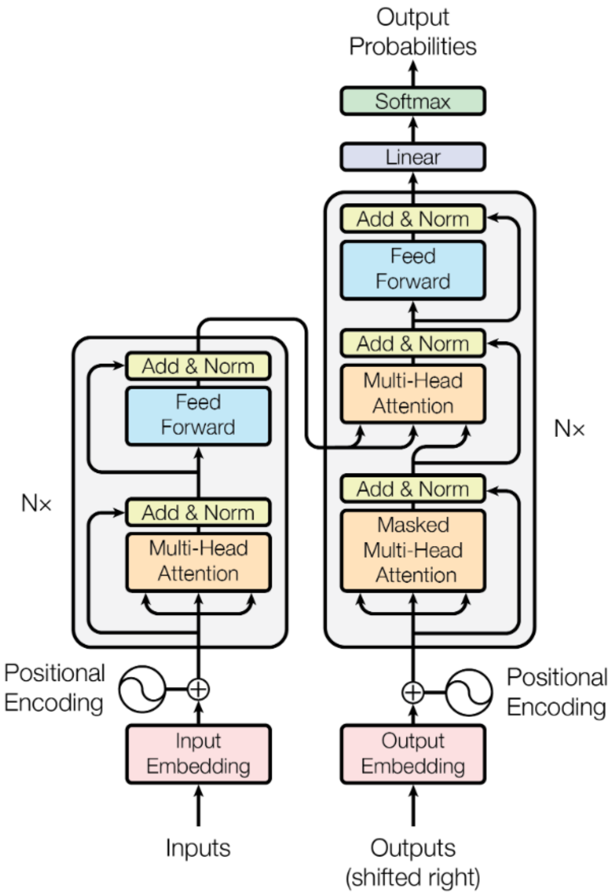






The Transformer I

- *Attention is all you need* by Vaswani, Ashish, et al.
  - NIPS 2017
  - https://arxiv.org/pdf/1706.03762.pdf
- Non-recurrent sequence-tosequence encoder-decoder model
- Task: machine translation with parallel corpus
- Predict each translated word
- Final cost/error function is standard cross-entropy error on top of a softmax classifier

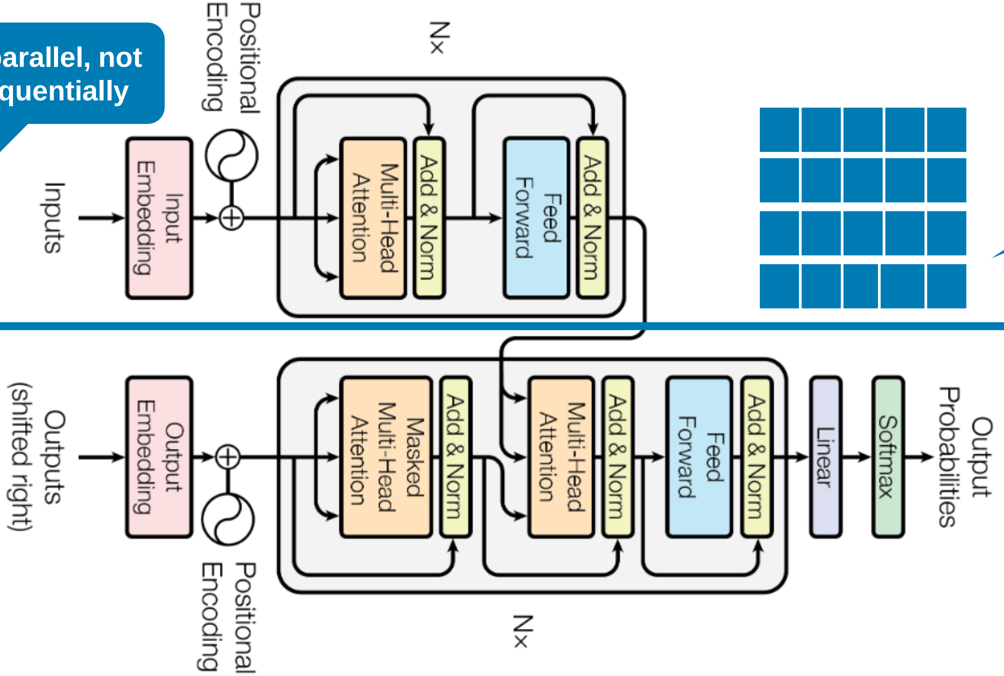


The Transformer II

Encoder

In parallel, not sequentially

Ich
gehe
nach
Hause



Embedding vectors

<start>
I
go

home

Decoder

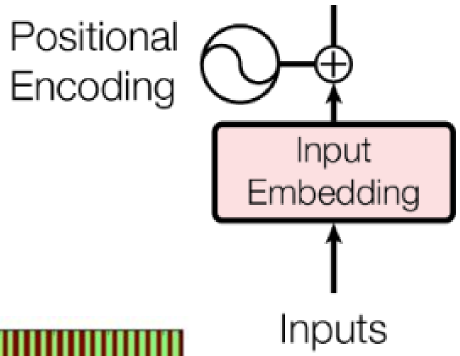
Encoder-Input I

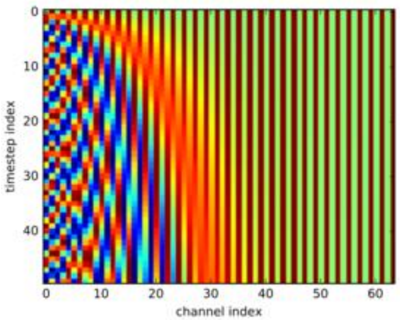
- Actual word representations are byte-pair encodings

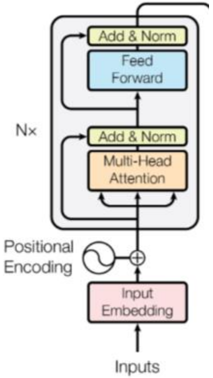
- **Positional encodings** are added so that same words at different locations have different overall representations and relative distance is considered:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

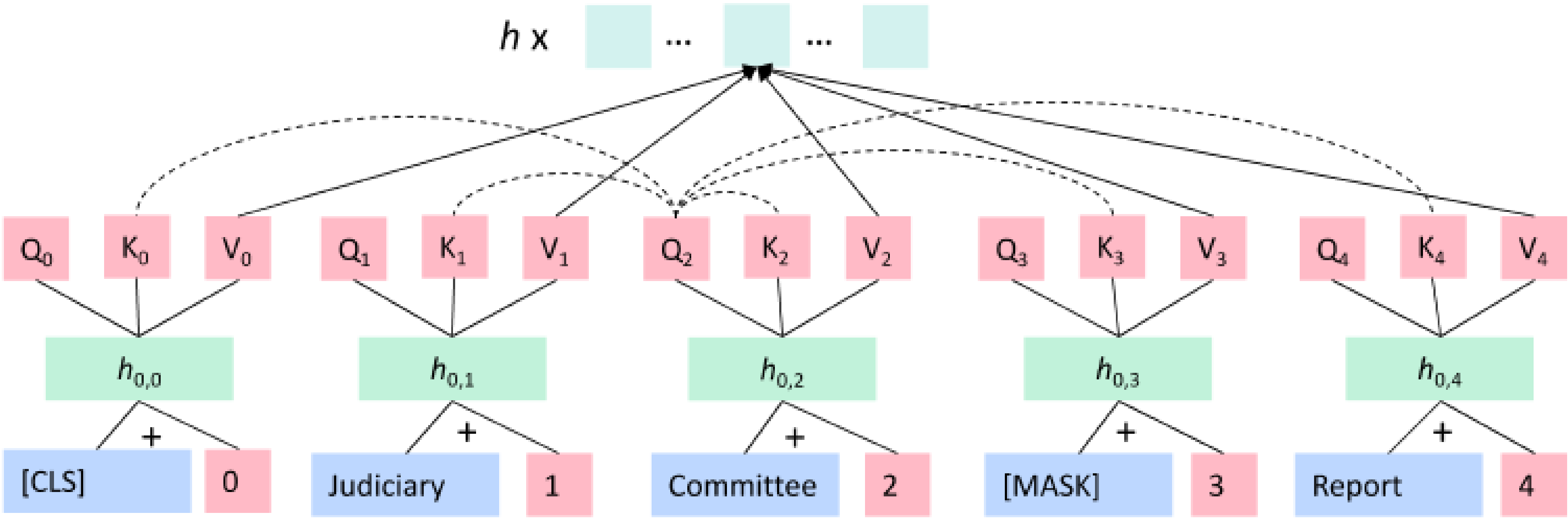
  – Can also be learned







Encoder-Input II



Dot-Product Attention

- Inputs: a query q and a set of key-value (k-v) pairs to an output
- Query, keys, values, and output are all vectors
- Output is weighted sum of values, where
  - Weight of each value is computed by an inner product of query and corresponding key
  - Queries and keys have same dimensionality d_k; values have dim d_v

$$A(q,K,V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i \rightarrow A(Q,K,V) = softmax(QK^T)V$$

Matrix notation for multiple queries q

- Self-Attention

Ich
gehe
nach
Hause

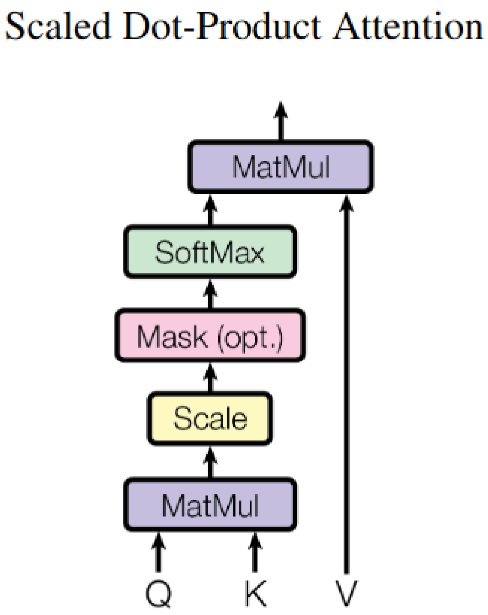
Focus on: Ich gehe nach Hause
Focus on: Ich gehe nach Hause
Focus on: Ich gehe nach Hause
Focus on: Ich gehe nach Hause

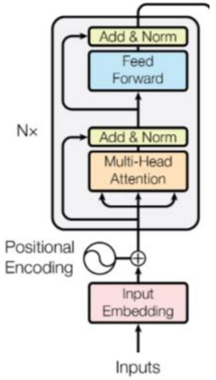
Scaled Dot-Product Attention

- Problem: As d_k gets large, the variance of $q^T k$ increases
  → some values inside the softmax get large
  → the softmax gets very peaked
  → hence its gradient gets smaller

- Solution: Scale by length of query/key vectors:

$$A(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{k_k}}\right)V$$





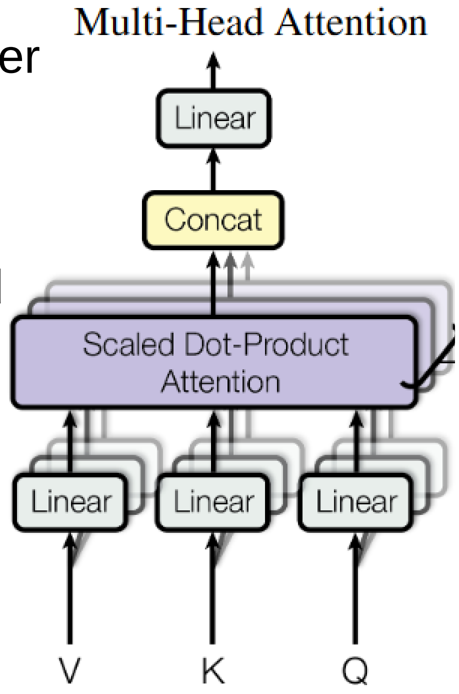
Self-Attention in an Encoder

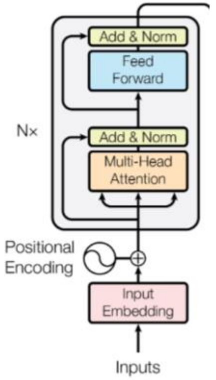
- The input word vectors are the queries, keys and values
- In other words: the word vectors themselves select each other
- Word vector stack = Q = K = V
- They're separated in the definition so you can do different things
  - For an NMT decoder, you can do queries from the output with K/V from the encoder

Multi-Head Attention

- Problem with simple self-attention:
  – Only one way for words to interact with one-another
- Solution: Multi-head attention
- First map Q, K, V into h=8 many lower dimensional spaces via W matrices
- Then apply attention, then concatenate outputs and pipe through linear layer

$$MultiHead(Q,K,V) = Concat(head_1, \dots, head_h)W^O$$
$$\text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

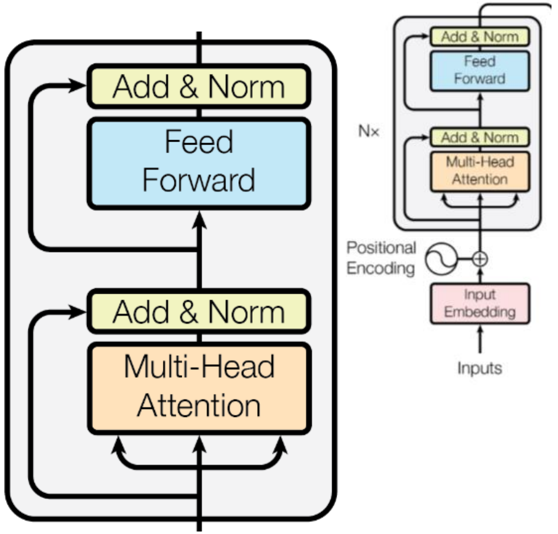




Complete Transformer Block

- Each block has two “sublayers”
  1. Multihead attention
  2. 2-layer feed-forward NNet (with ReLU)

- Each of these two steps also has:
  – Residual (short-circuit) connection and LayerNorm
  – LayerNorm changes input features to have mean 0 and variance 1 per layer (and adds two more parameters)

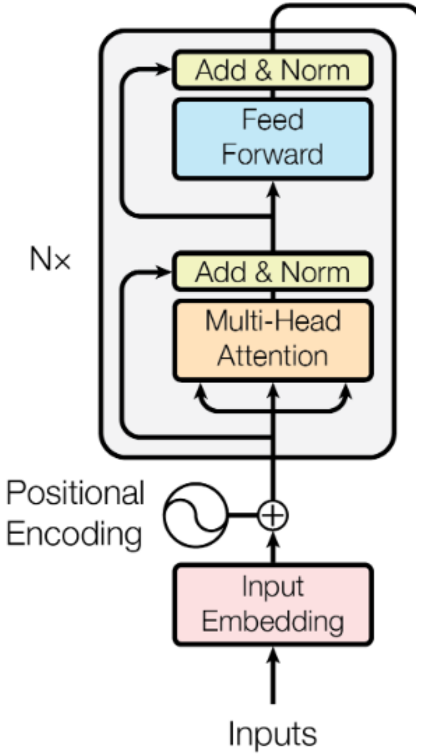


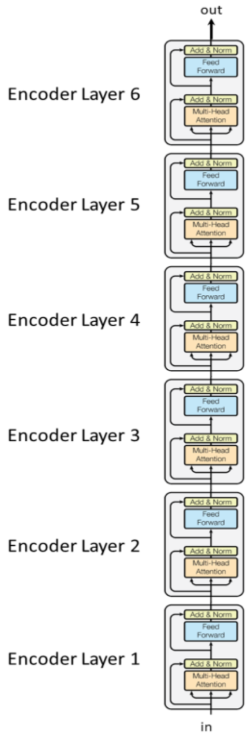
$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \qquad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)^2} \qquad h_i = f\left(\frac{g_i}{\sigma_i}\left(a_i - \mu_i\right) + b_i\right)$$

Layer Normalization by Ba, Kiros and Hinton. https://arxiv.org/pdf/1607.06450.pdf

Complete Encoder

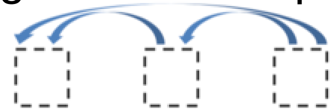
- Blocks are repeated 6 or more times
  - (in vertical stack)
  - Inputs are Q, K and V of previous layer





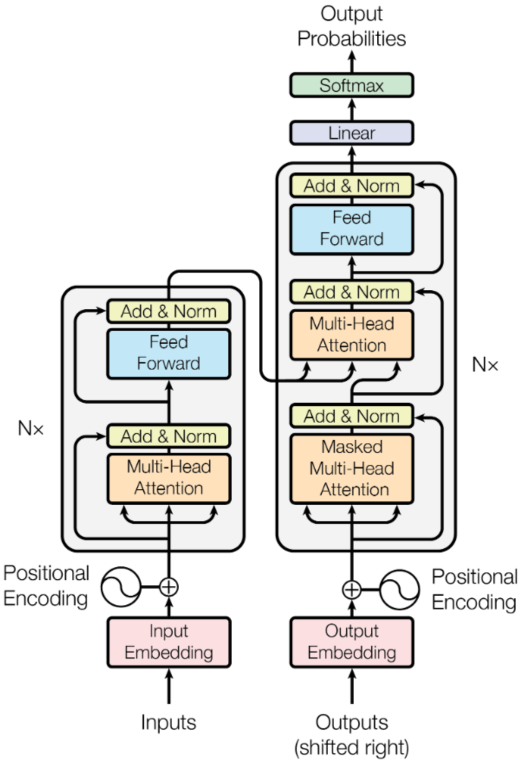
Complete Decoder

- 2 sublayers change in decoder
- Masked decoder self-attention on previously generated outputs:



- Encoder-Decoder Attention, where queries come from previous decoder layer and keys and values come from output of encoder

- Blocks repeated 6 times also



Experimental Results for MT

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

- The Transformer achieves better BLEU scores than previous state-of-the-art models on the EN-DE and EN-FR newstest2014 tests at **a fraction of the training cost**

Some Performance Numbers: LM on WikiText-103

Model	# Params	Perplexity
Grave et al. (2016) – LSTM		48.7
Grave et al. (2016) – LSTM with cache		40.8
4-layer QRNN (Merity et al. 2018)	151M	33.0
LSTM + Hebbian + Cache + MbPA (Rae et al.)	151M	29.2
Transformer-XL Large (Dai et al. 2019)	**257M**	**18.3**
GPT-2 Large* (Radford et al. 2019)	1.5B	17.5

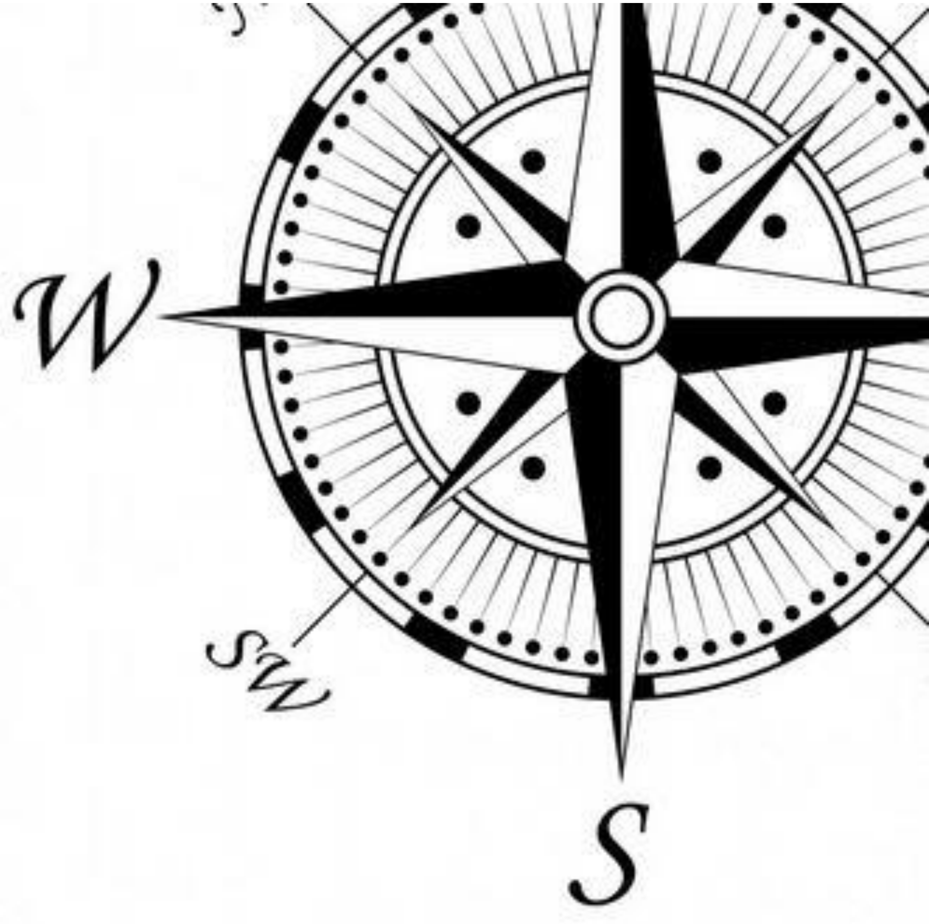
Transformer

- Go through the Tensorflow transformer tutorial
  - https://www.tensorflow.org/tutorials/text/transformer

Start 5 4 3 2 1 Ende

Topics Today

1. Transformer
2. **BERT**
3. Current Developments



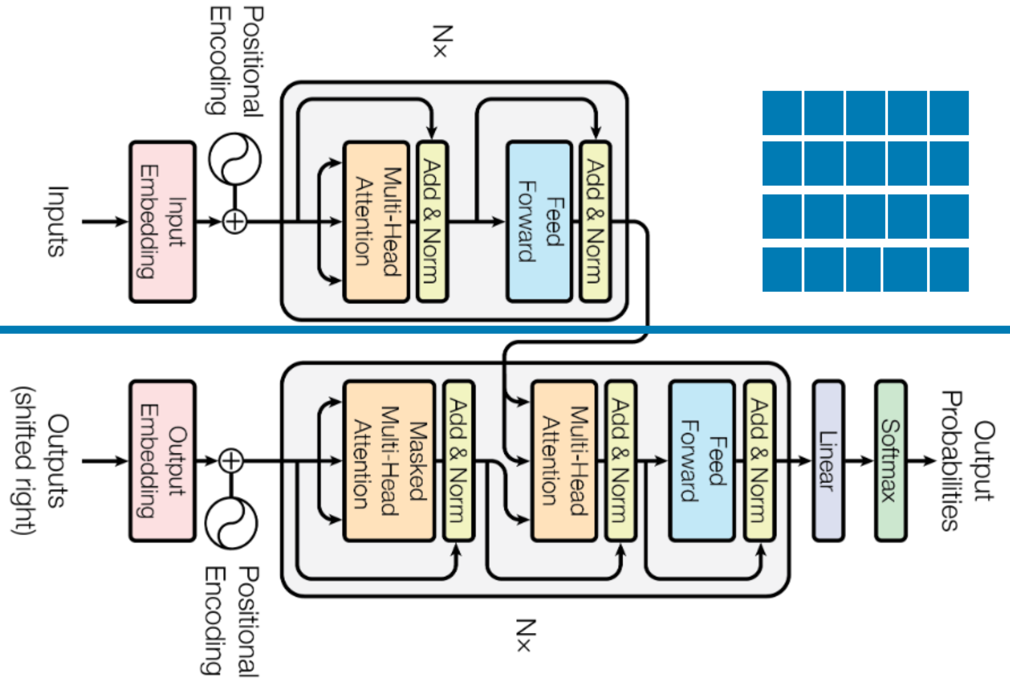
Timeline

ELMo	ULMfit	GPT	BERT	GPT-2	XL-Net, ERNIE, Grover, RoBERTa, T5, GPT-3, Big BIRD
Oct 2017	Jan 2018	June 2018	Oct 2018	Feb 2019	Juli 2019 ++

From Transformer To BERT

Encoder

Ich
gehe
nach
Hause



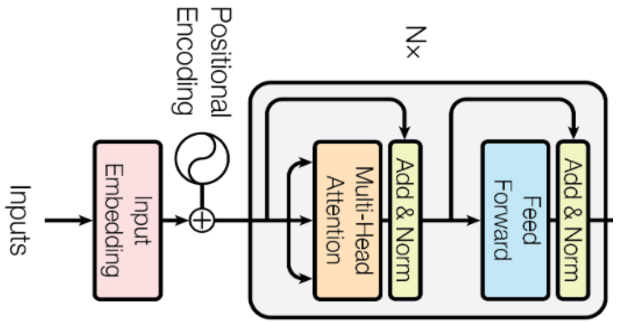
Embedding vectors

<start>
I
go

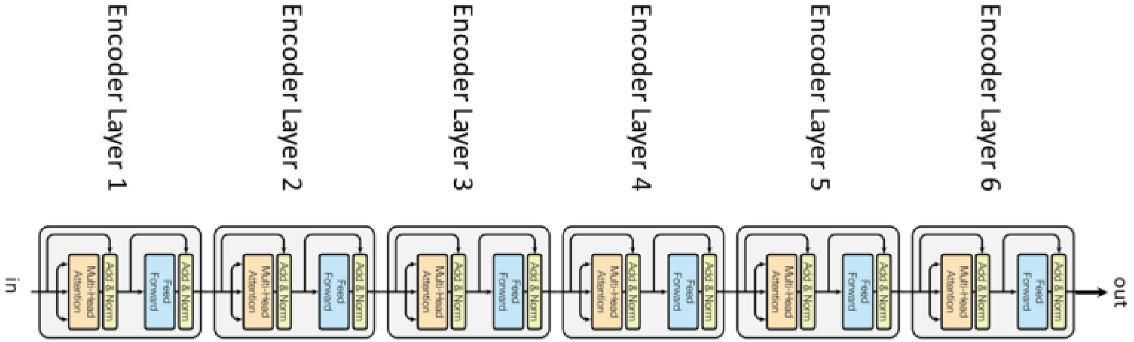
home

Decoder

From Transformer to BERT



- Self-Attention
  → no locality bias
  – Distance does not matter for context
- Single multiplication per layer
  → very efficient on GPUs/TPUs

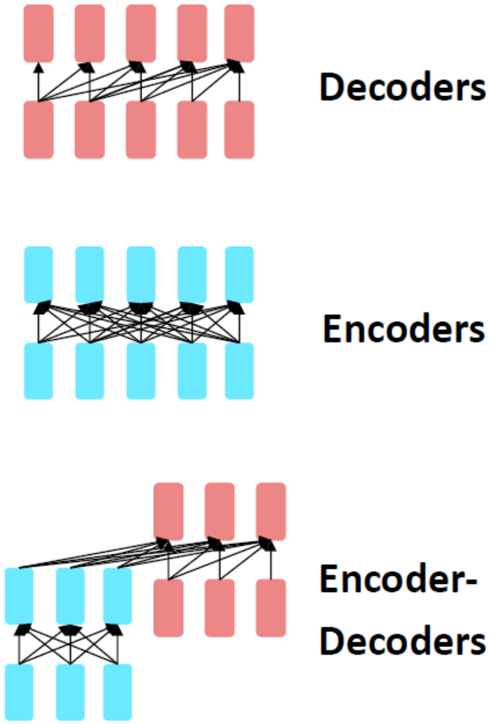


GPT = stacked decoders

BERT – GPT – Transformers

- Decoders only
  - Language models
  - Very good for generating
  - Examples
    - GPT, GPT2, GPT-3, LaMDA
- Encoders only
  - Gets bidirectional context
  - Can condition on future
  - Examples
    - BERT and its many variants, e.g., RoBERTa
- Encoder-Decoders
  - Combines encoders and decoders
  - Examples
    - Transformer, T5, Meena

      https://ai.googleblog.com/2020/01/towards-conversational-agent-that-can.html



BERT

- **Bidirectional Encoder Representations from Transformers**
  – Devlin, Chang, Lee, Toutanova (2018)

- Pre-training of Deep Bidirectional Transformers for Language Understanding, which is then **fine-tuned for a task**

- Want: truly bidirectional information flow without leakage in a deep model

https://pixy.org/src/425/4254306.png

BERT Pre-Training

- Two tasks
  1. Learn to solve cloze task
     - Masked Language Model (MLM)
     - 15% of words of the training texts are blanked out and need to be predicted:

store ↑ gallon ↑

the man went to the [MASK] to buy a [MASK] of milk

  2. Predict the next sentence
     - Next Sentence Prediction (NSP)

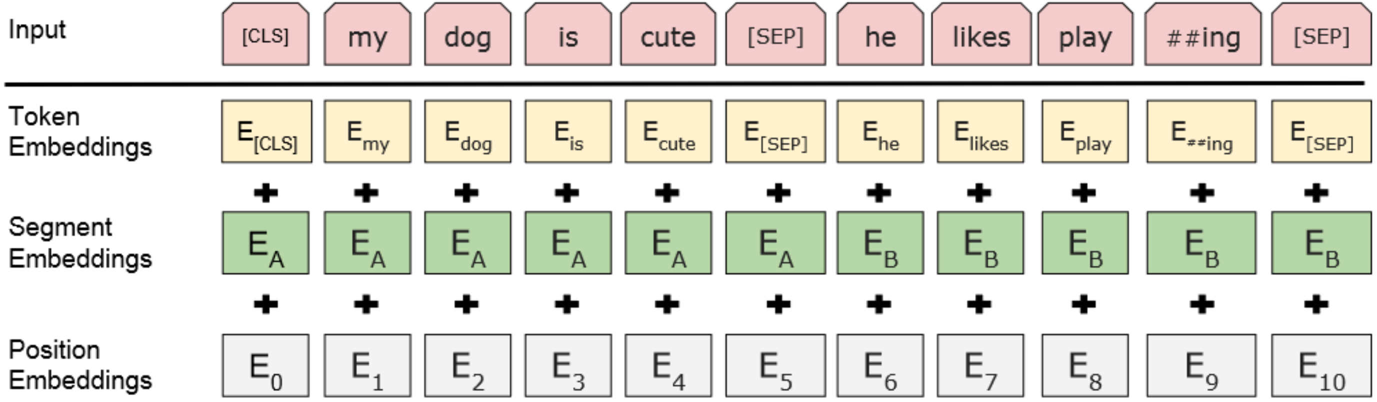
       A: The weather is nice.

       B: We go for a swim. → Yes, B is next sentence after A

https://pixy.org/src/425/4254306.png

BERT Pre-Training: Eingabe

- Token embeddings are word pieces
- Learned segmented embedding represents each sentence

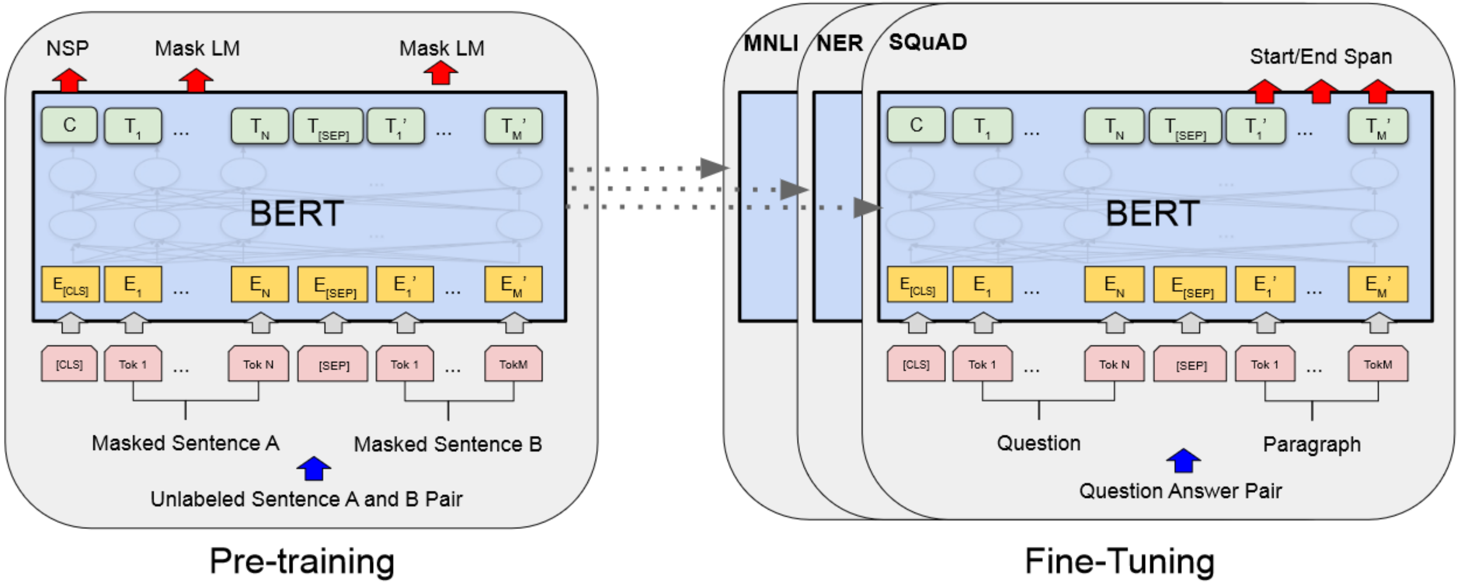


Wordpiece embeddings (Vocabulary size= 30k)

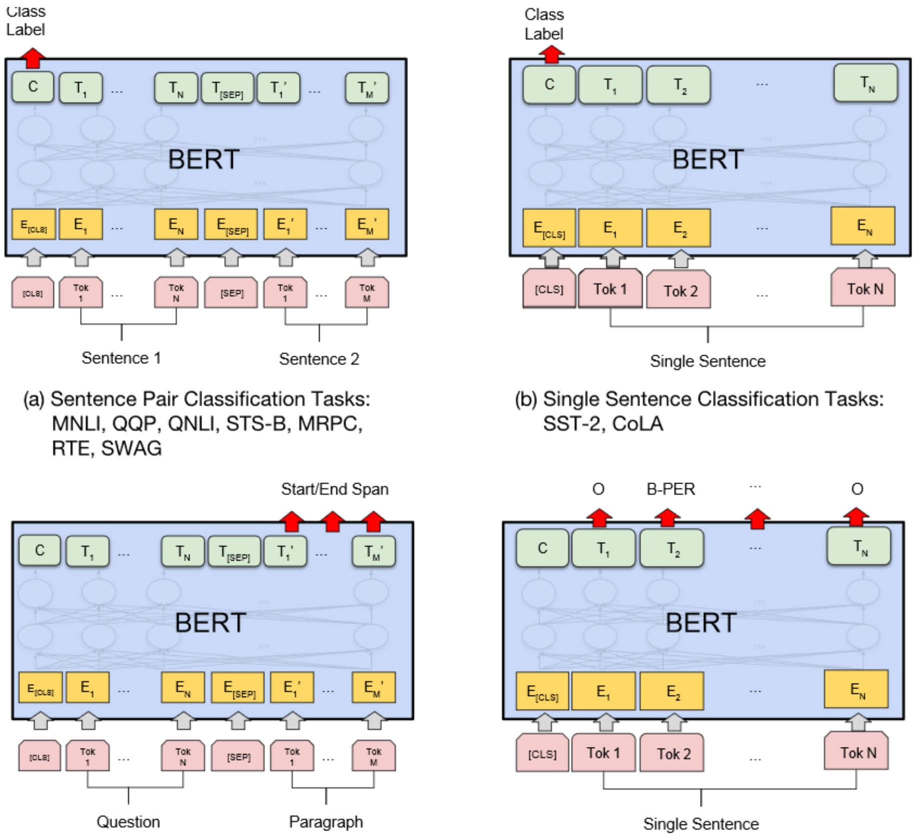
No sequencial input (cf. RNN), therefore position information necessary

BERT Fine-Tuning I

- Simply learn a classifier built on the top layer for each task that you fine tune for, e.g. QA
  - Exchange last layer and learn (only) their weights in a supervised manner
  - Fine-Tuning of BERT model based on selected task



BERT Fine-Tuning II



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

BERT Implementation

- Trained on Wikipedia (2.5B words) + books corpus (800M words)

- 2 different sizes:
  - BERT-Base: 12-layer, 768-hidden, 12-head
  - BERT-Large: 24-layer, 1024-hidden, 16-head

110M parameters

340M parameters

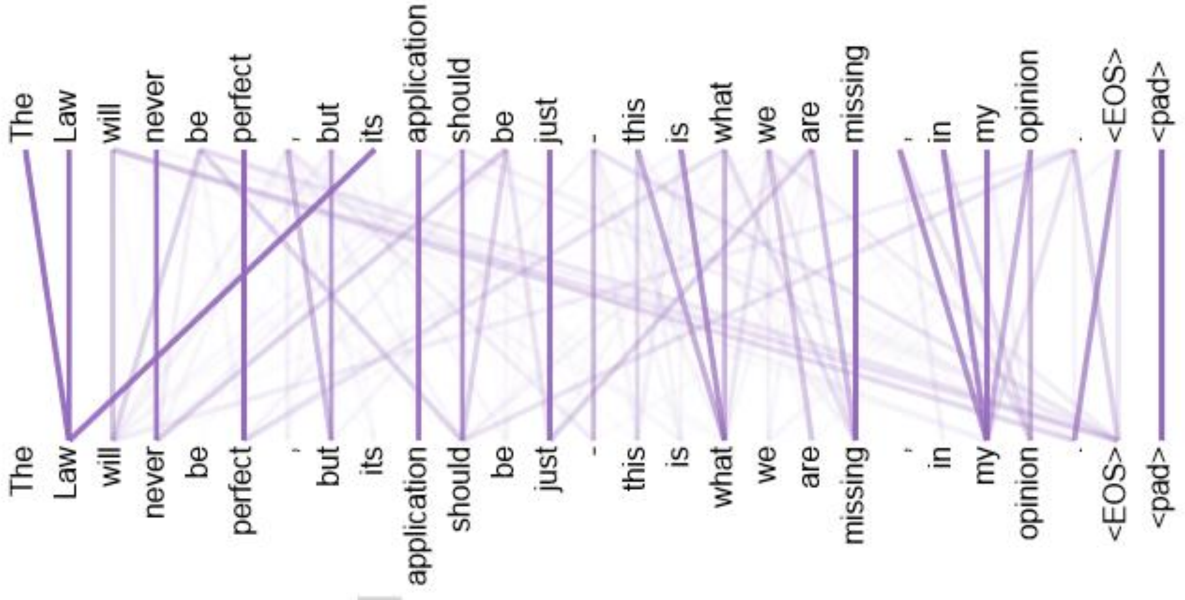
- Pretraining is expensive and impractical on a single GPU
  - BERT was pretrained with 64 TPU chips for a total of 4 days.

- Finetuning is practical and common on a single GPU
  - "Pretrain once, finetune many times."

https://pixy.org/src/425/4254306.png

Visualization of Attention

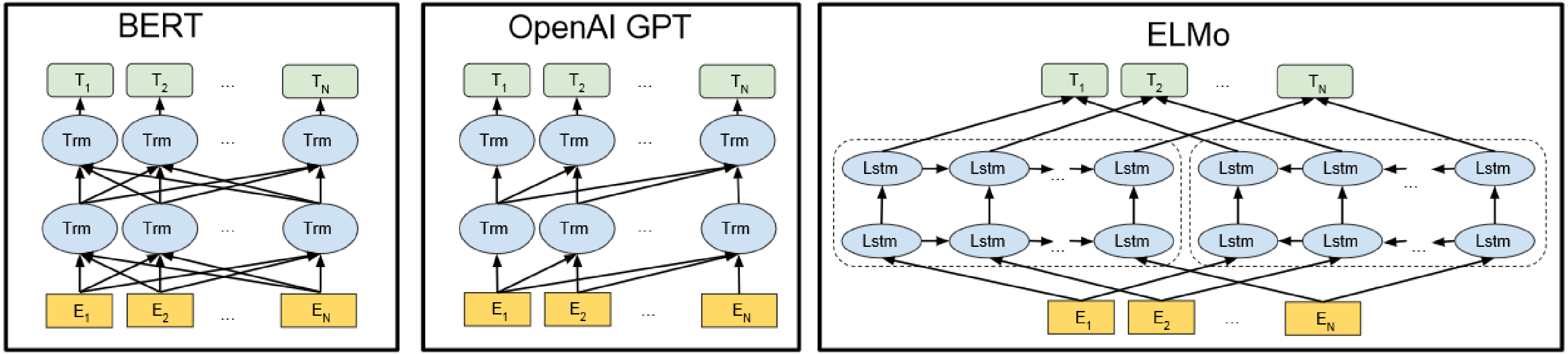
- Implicit resolution of anaphers
  - Words start to pay attention to other words in sensible ways



5th layer; Attention-Head 5

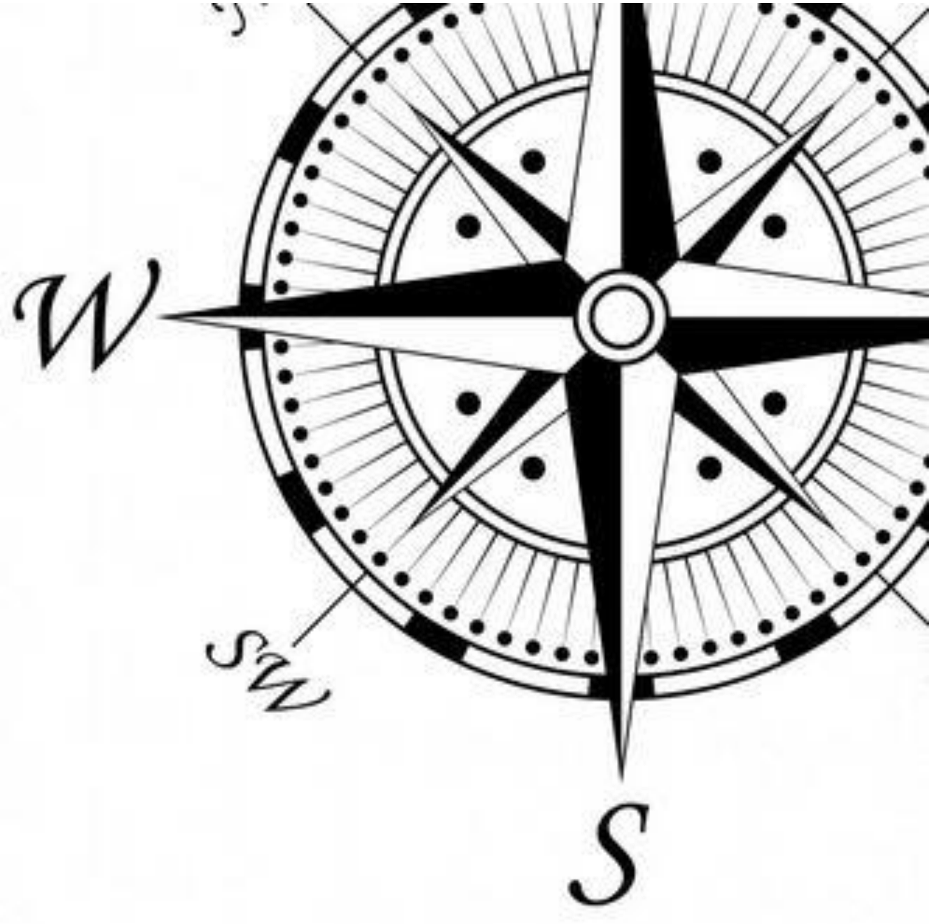
Summary: BERT – GPT – ELMo

- https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html



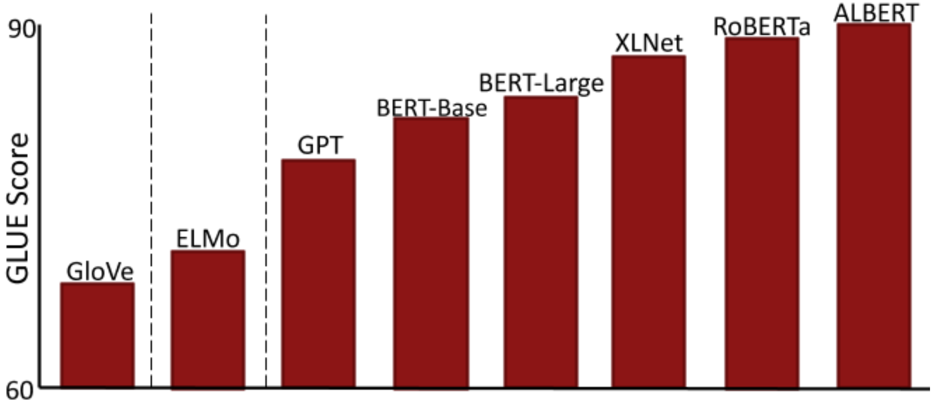
Topics Today

1. Transformer
2. BERT
3. **Current Developments**



GLUE Benchmark: Results over Time

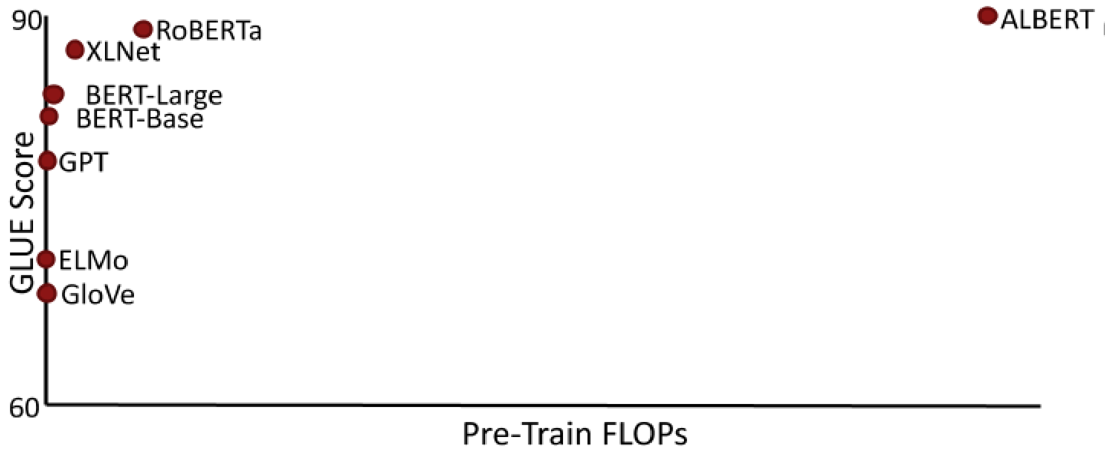
- General Language Understanding Evaluation
  - Within 2 years, error dropped by 2/3
  - “Superhuman” Performance



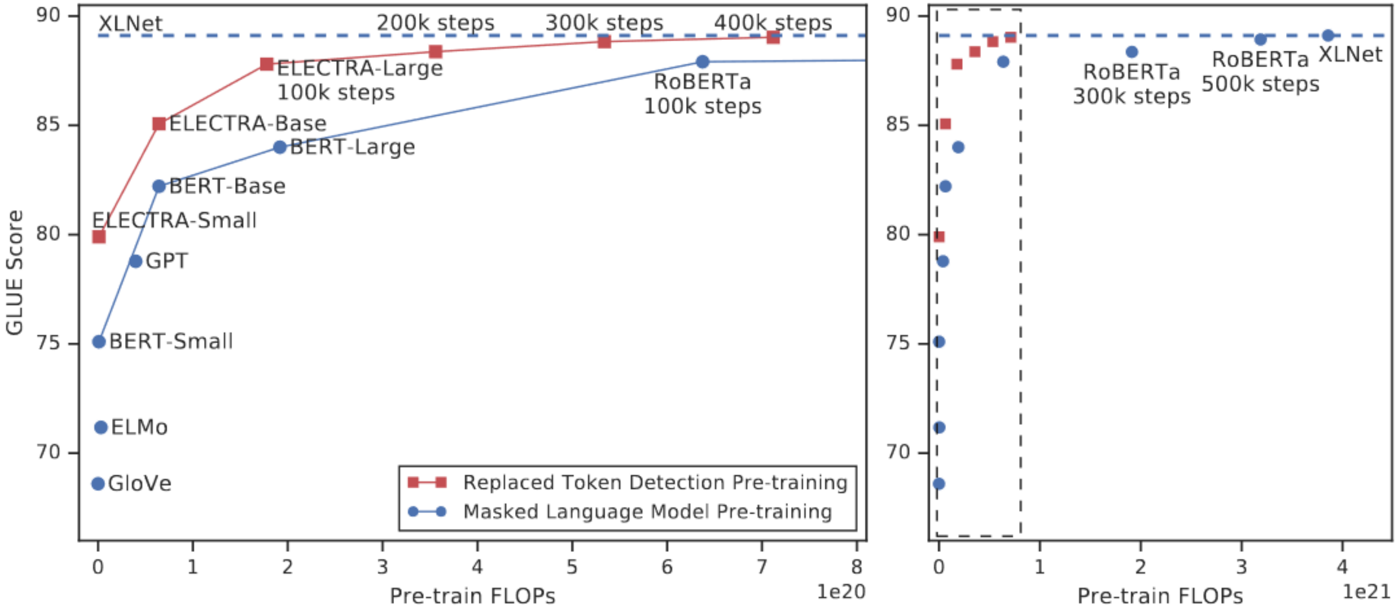
- Since 2018 we have strongly performing, deep, generic, pre-trained, neural network stacks for NLP that you can just load – in the same way vision has had 5 years earlier (ResNet, etc.)!

GLUE Benchmark: Compute Power

- BERT-Large uses 60x more compute than ELMo
- RoBERTa uses 16x more compute than BERT-Large
- ALBERT uses 10x more compute than RoBERTa



Results: GLUE-Score vs. Compute Power



Learning Goals for this Chapter

- Understand transformers
- Know how BERT works and how to use it
- Understand current developments for word embeddings/language models

- Relevant chapters:
  – S9 (2021): https://www.youtube.com/watch?v=ptuGllU5SQQ
  – S10 (2021): https://www.youtube.com/watch?v=j9AcEI98C0o

Literature I

- Overview paper: Smith, Noah (2019) "Contextual word representations: A contextual introduction“
  - https://arxiv.org/abs/1902.06006
- Blogpost GPT-3: The Dream Machine in the Real World
  - https://towardsdatascience.com/gpt3-the-dream-machine-in-real-world-c99592d4842f
- Blogpost The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)
  - http://jalammar.github.io/illustrated-bert/
- Tensorflow Python notebook: Annotated transformer code
  - https://www.tensorflow.org/tutorials/text/transformer

Literature II

- Vaswani, Ashish, et al. "Attention is all you need.“
  - *Advances in neural information processing systems*. 2017.
- Devlin, Jacob, et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.“
  - *Proceedings of NAACL*. 2019.