IA-312 - Attention mechanism, Transformers and Large Scale NLP

Matthieu Labeau

matthieu.labeau@telecom-paris.fr

Attention Mechanism

Seq2seq models: with Alignment

Neural Machine Translation by Jointly Learning to Align and Translate (Badhanau et al, 2014)

$\rightarrow$ **Idea**: during decoding, look at the input sequence and focus on important words

Intuition: when decoding "*réseau*", we should focus on "*network*"

$$f(\text{réseau}, [\text{This}, \text{network}, \text{is}, \text{complex}]) \sim \text{network}$$

Seq2seq models: with Alignment

Neural Machine Translation by Jointly Learning to Align and Translate (Badhanau et al, 2014)

→ **Idea**: during decoding, look at the input sequence and focus on important words

Intuition: when decoding "*réseau*", we should focus on "*network*"

Seq2seq models: with Alignment

Neural Machine Translation by Jointly Learning to Align and Translate (Badhanau et al, 2014)

$\rightarrow$ **Idea**: during decoding, look at the input sequence and focus on important words

Intuition: when decoding "*réseau*", we should focus on "*network*"

$$f(\quad\quad\quad\quad\quad\quad,\quad) = [\quad\quad,\quad\quad,\quad\quad,\quad\quad]$$

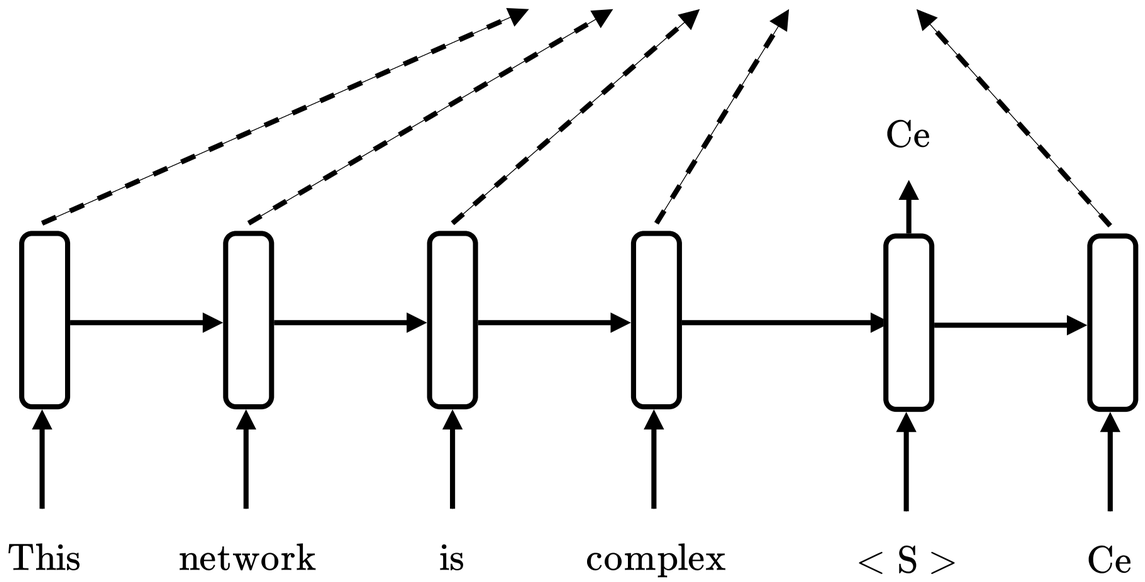
Seq2seq models: with Alignment

Neural Machine Translation by Jointly Learning to Align and Translate (Badhanau et al, 2014)

$\rightarrow$ **Idea**: during decoding, look at the input sequence and focus on important words

Intuition: when decoding "*réseau*", we should focus on "*network*"

$$f(\quad\quad\quad\quad\quad\quad,\quad) = [\quad\quad,\quad\quad,\quad\quad,\quad\quad]$$

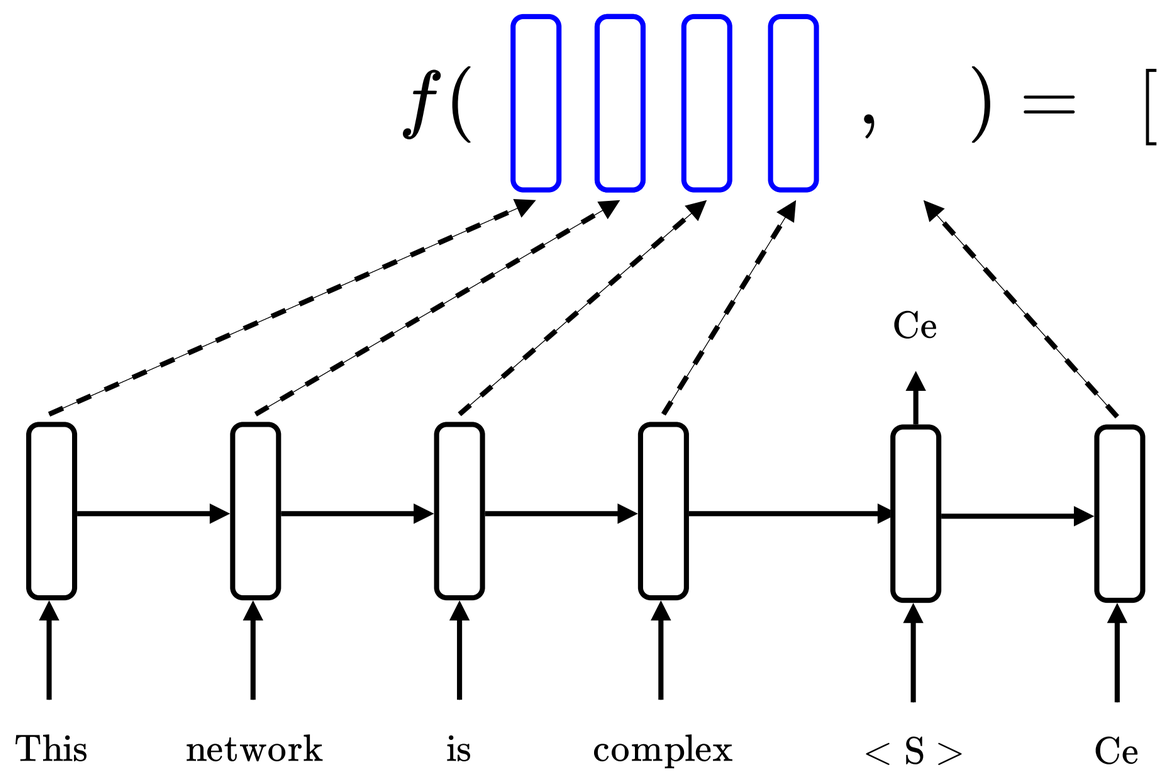


Seq2seq models: with Alignment

Neural Machine Translation by Jointly Learning to Align and Translate (Badhanau et al, 2014)

→ **Idea**: during decoding, look at the input sequence and focus on important words

Intuition: when decoding "*réseau*", we should focus on "*network*"



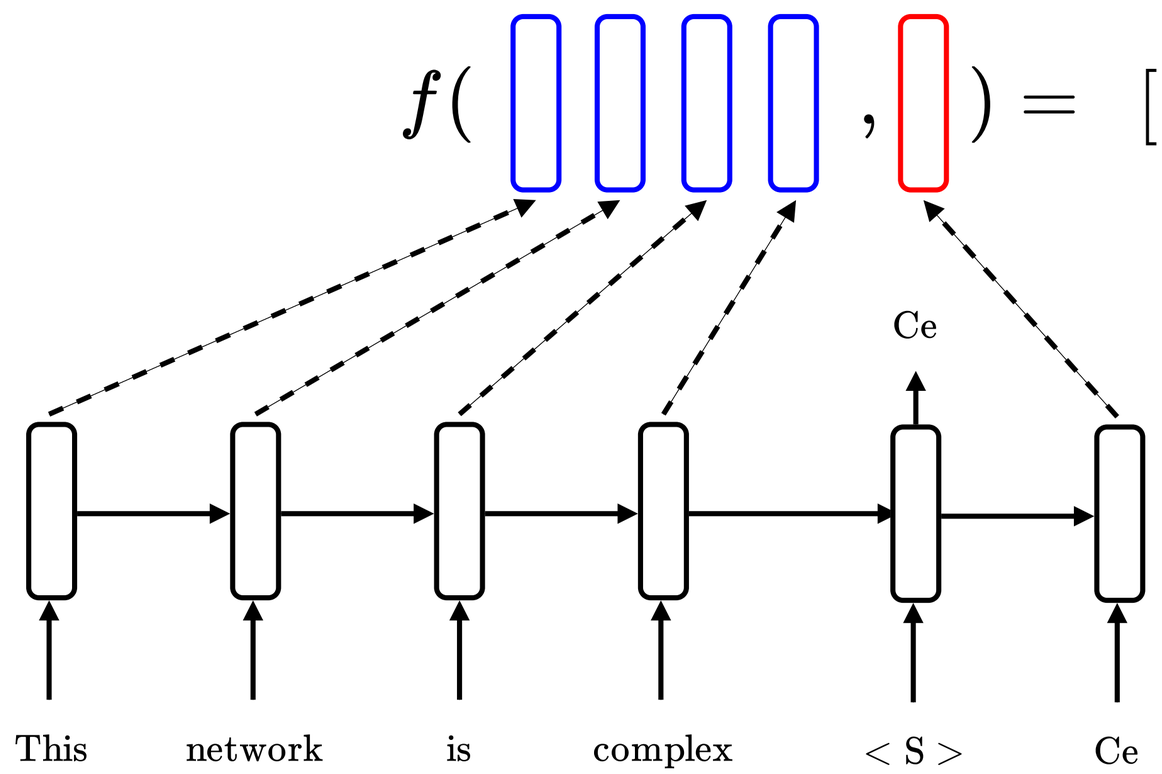
- Encoder states are keys

Seq2seq models: with Alignment

Neural Machine Translation by Jointly Learning to Align and Translate (Badhanau et al, 2014)

→ **Idea**: during decoding, look at the input sequence and focus on important words

Intuition: when decoding "*réseau*", we should focus on "*network*"



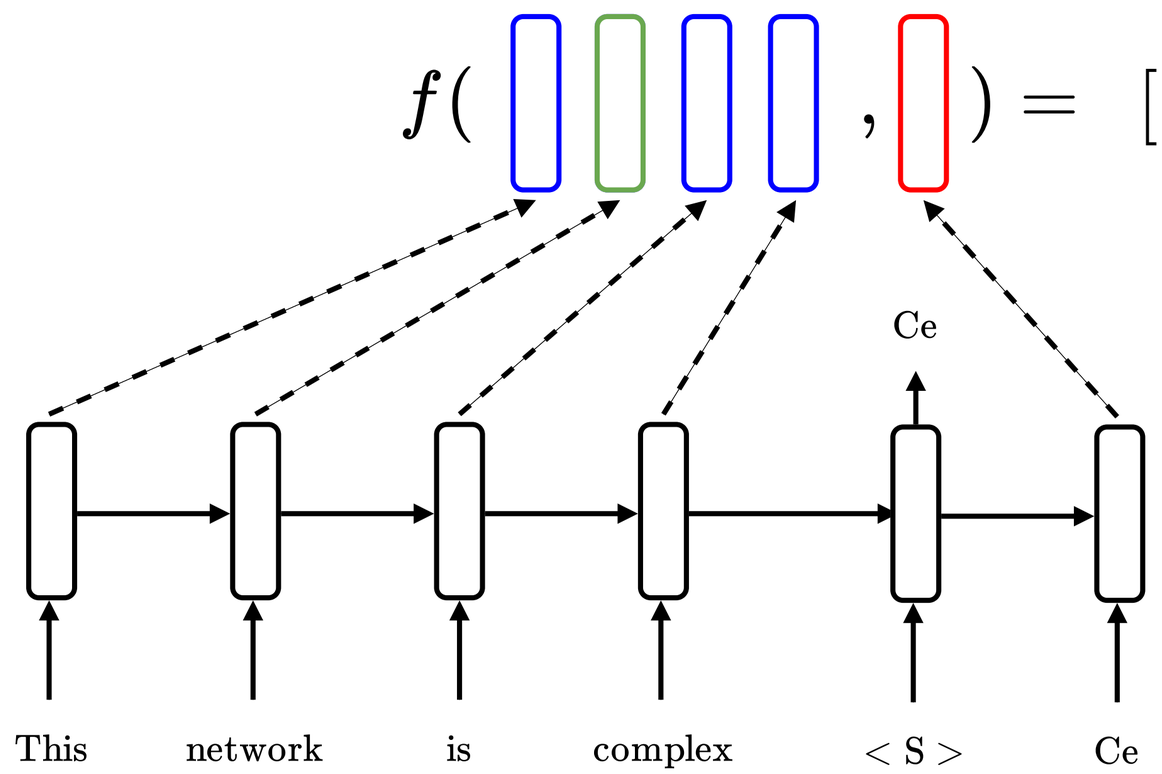
- Encoder states are keys
- The current decoder state is the query

Seq2seq models: with Alignment

Neural Machine Translation by Jointly Learning to Align and Translate (Badhanau et al, 2014)

$\rightarrow$ **Idea**: during decoding, look at the input sequence and focus on important words

Intuition: when decoding "*réseau*", we should focus on "*network*"



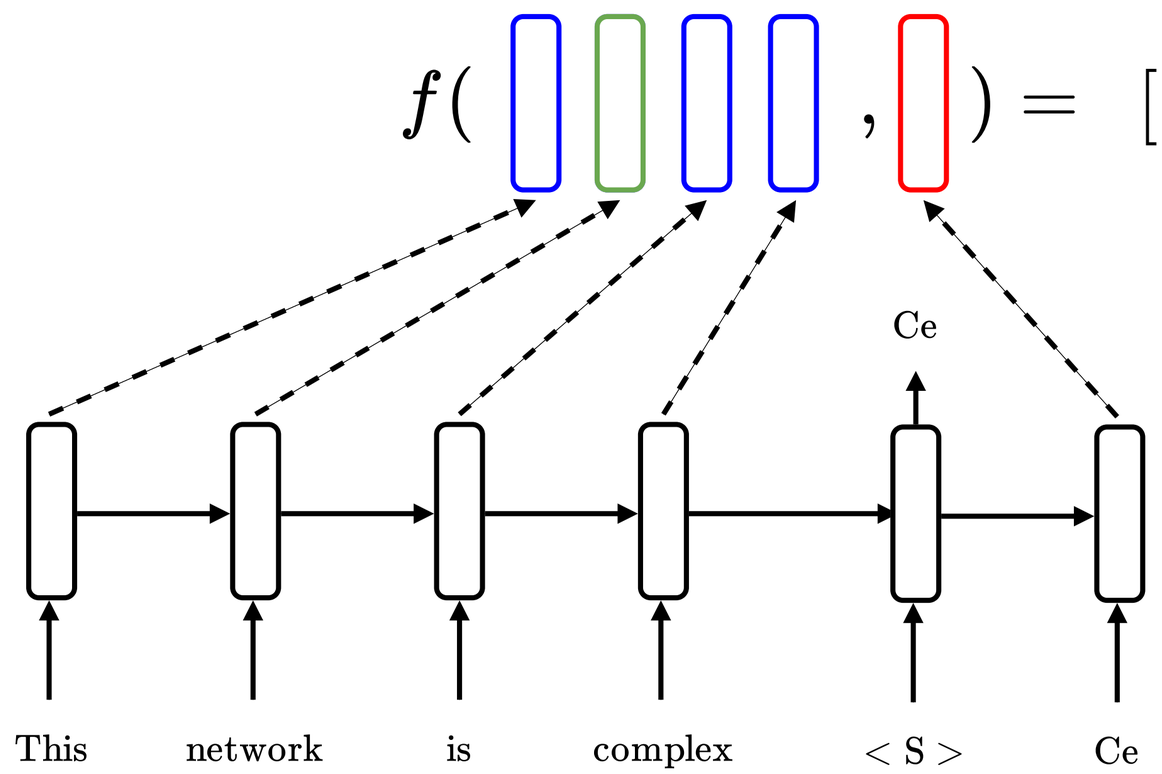
- Encoder states are keys
- The current decoder state is the query
- The state giving the maximum weight among the encoder states (now called values) is used for decoding

Seq2seq models: with Alignment

Neural Machine Translation by Jointly Learning to Align and Translate (Badhanau et al, 2014)

→ **Idea**: during decoding, look at the input sequence and focus on important words

Intuition: when decoding "*réseau*", we should focus on "*network*"

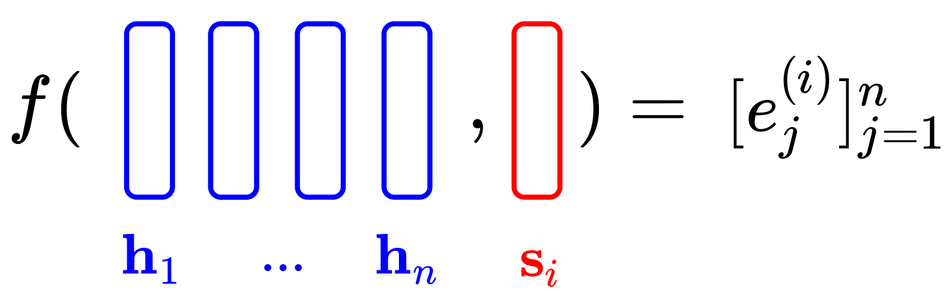


- Encoder states are keys
- The current decoder state is the query
- The state giving the maximum weight among the encoder states (now called values) is used for decoding

...but we want this process to be **differentiable** and **trainable end-to-end** !

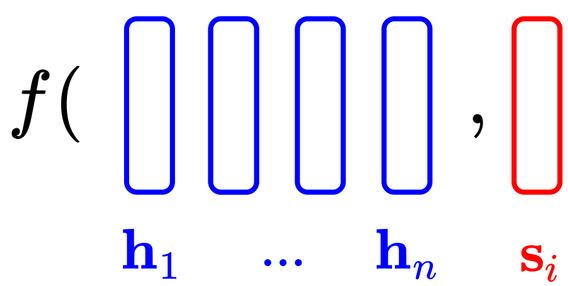
Seq2seq models: Attention

Ideas: use a **simple similarity measure** between keys and query and a **softmax** instead of an argmax to select the value !



Seq2seq models: Attention

Ideas: use a **simple similarity measure** between keys and query and a **softmax** instead of an argmax to select the value !



$$\mathbf{e}_j^{(i)} = s(\mathbf{s}_i, \mathbf{h}_j)$$

Seq2seq models: Attention

Ideas: use a **simple similarity measure** between keys and query and a **softmax** instead of an argmax to select the value !

$$f(\ \square\ \square\ \square\ \square\ ,\ \square\ ) = [e_j^{(i)}]_{j=1}^{n} \qquad \mathbf{e}_j^{(i)} = s(\mathbf{s}_i, \mathbf{h}_j)$$

$\mathbf{h}_1 \quad \dots \quad \mathbf{h}_n \quad \mathbf{s}_i$

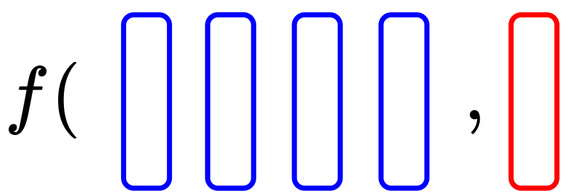
- Attention weights are computed with a softmax

$$\downarrow$$

$$\mathbf{a}_j^{(i)} = \mathrm{softmax}(\mathbf{e}_j^{(i)})$$

Seq2seq models: Attention

Ideas: use a **simple similarity measure** between keys and query and a **softmax** instead of an argmax to select the value !



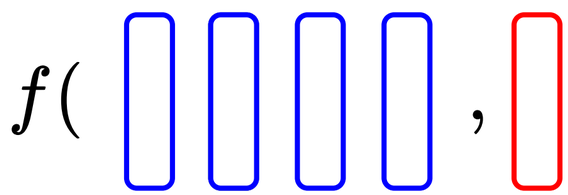
$$\mathbf{e}_j^{(i)} = s(\mathbf{s}_i, \mathbf{h}_j)$$

- Attention weights are computed with a softmax

$$\mathbf{a}_j^{(i)} = \mathrm{softmax}(\mathbf{e}_j^{(i)}) = \frac{\exp(f(\mathbf{s}_i, \mathbf{h}_j))}{\sum_{l=1}^{n} \exp(f(\mathbf{s}_i, \mathbf{h}_l))}$$

Seq2seq models: Attention

Ideas: use a **simple similarity measure** between keys and query and a **softmax** instead of an argmax to select the value !



$$\mathbf{e}_j^{(i)} = s(\mathbf{s}_i, \mathbf{h}_j)$$

- Attention weights are computed with a softmax

$$\downarrow$$

$$\mathbf{a}_j^{(i)} = \mathrm{softmax}(\mathbf{e}_j^{(i)}) = \frac{\exp(f(\mathbf{s}_i, \mathbf{h}_j))}{\sum_{l=1}^n \exp(f(\mathbf{s}_i, \mathbf{h}_l))}$$

- The output is obtained through weighted sum of the values

$$\downarrow$$

$$\mathbf{z}_i = \sum_{j=1}^n a_j^{(i)} \mathbf{h}_j$$

Seq2seq models: Attention

Ideas: use a **simple similarity measure** between keys and query and a **softmax** instead of an argmax to select the value !

$$f\left(\begin{array}{cccc}\square & \square & \square & \square\end{array}, \square\right) = [e_j^{(i)}]_{j=1}^n \qquad \mathbf{e}_j^{(i)} = s(\mathbf{s}_i, \mathbf{h}_j)$$

$\mathbf{h}_1 \quad \dots \quad \mathbf{h}_n \quad \mathbf{s}_i$

- Attention weights are computed with a softmax

$$\downarrow$$

$$\mathbf{a}_j^{(i)} = \mathrm{softmax}(\mathbf{e}_j^{(i)}) = \frac{\exp(f(\mathbf{s}_i, \mathbf{h}_j))}{\sum_{l=1}^n \exp(f(\mathbf{s}_i, \mathbf{h}_l))}$$

- The output is obtained through weighted sum of the values

$$\downarrow$$

$$\mathbf{z}_i = \sum_{j=1}^n a_j^{(i)} \mathbf{h}_j$$

...and all of this is **differentiable** and fast to compute !

Attention for NMT: Assessment

Neural Machine Translation by Jointly Learning to Align and Translate (Bahdanau et al, 2014)

- Attention largely improves NMT performance !

Attention for NMT: Assessment

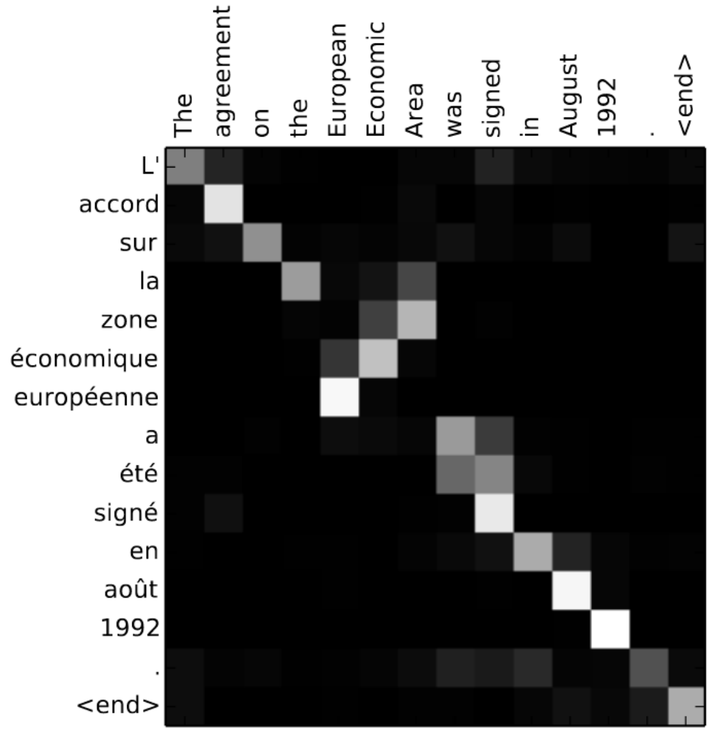
Neural Machine Translation by Jointly Learning to Align and Translate (Bahdanau et al, 2014)

- Attention largely improves NMT performance !
- Besides solving the information bottleneck problem, it allows the model to reflect **far longer dependencies**.

Attention for NMT: Assessment

Neural Machine Translation by Jointly Learning to Align and Translate (Bahdanau et al, 2014)

- Attention largely improves NMT performance !
- Besides solving the information bottleneck problem, it allows the model to reflect **far longer dependencies**.
- Attention gives **soft alignement** on generated translations:



Attention: other uses

- Similarly, **Attention** is a general technique, that can be applied within other models and to many other tasks, allowing to obtain a *fixed sized* representation from an *arbitrary number* of representations
  - You can use it in any architecture, with any set of inputs !

Task: Hotel location

you get what you pay for . not the cleanest rooms but bed was clean and so was bathroom . bring your own towels though as very thin . service was excellent , let us book in at 8:30am ! **for location and price , this ca n't be beaten** , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Task: Hotel cleanliness

you get what you pay for . **not the cleanest rooms but bed was clean and so was bathroom** . bring your own towels though as very thin . service was excellent , let us book in at 8:30am ! for location and price , this ca n't be beaten , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Task: Hotel service

you get what you pay for . not the cleanest rooms but bed was clean and so was bathroom . bring your own towels though as very thin . **service was excellent** , let us book in at 8:30am ! for location and price , this ca n't be beaten , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Attention: other uses

- Similarly, **Attention** is a general technique, that can be applied within other models and to many other tasks, allowing to obtain a *fixed sized* representation from an *arbitrary number* of representations
  - You can use it in any architecture, with any set of inputs !

Task: Hotel location

you get what you pay for . not the cleanest rooms but bed was clean and so was bathroom . bring your own towels though as very thin . service was excellent , let us book in at 8:30am ! **for location and price , this ca n't be beaten** , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Task: Hotel cleanliness

you get what you pay for . **not the cleanest rooms but bed was clean and so was bathroom** . bring your own towels though as very thin . service was excellent , let us book in at 8:30am ! for location and price , this ca n't be beaten , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Task: Hotel service

you get what you pay for . not the cleanest rooms but bed was clean and so was bathroom . bring your own towels though as very thin . **service was excellent** , let us book in at 8:30am ! for location and price , this ca n't be beaten , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Attention in Natural Language Processing (Galassi et al, 2019)

Attention: other uses

- Similarly, **Attention** is a general technique, that can be applied within other models and to many other tasks, allowing to obtain a *fixed sized* representation from an *arbitrary number* of representations
  - You can use it in any architecture, with any set of inputs !

Task: Hotel location

you get what you pay for . not the cleanest rooms but bed was clean and so was bathroom . bring your own towels though as very thin . service was excellent , let us book in at 8:30am ! **for location and price , this ca n't be beaten** , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Task: Hotel cleanliness

you get what you pay for . **not the cleanest rooms but bed was clean and so was bathroom** . bring your own towels though as very thin . service was excellent , let us book in at 8:30am ! for location and price , this ca n't be beaten , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Task: Hotel service

you get what you pay for . not the cleanest rooms but bed was clean and so was bathroom . bring your own towels though as very thin . **service was excellent** , let us book in at 8:30am ! for location and price , this ca n't be beaten , but it is cheap for a reason . if you come expecting the hilton , then book the hilton ! for uk travellers , think of a blackpool b&b.

Attention in Natural Language Processing (Galassi et al, 2019)

- Overall: **NMT** has been a **catalyst** for the development of DL techniques for NLP - and Seq2seq/Attention have been applied a lot

What's next ?

Issues with **recurrent** encoders and decoders:

- Relationships between words do not necessarily follow the linear order
- GPUs can perform parallel computations, but recurrent models need to compute states in order, successively

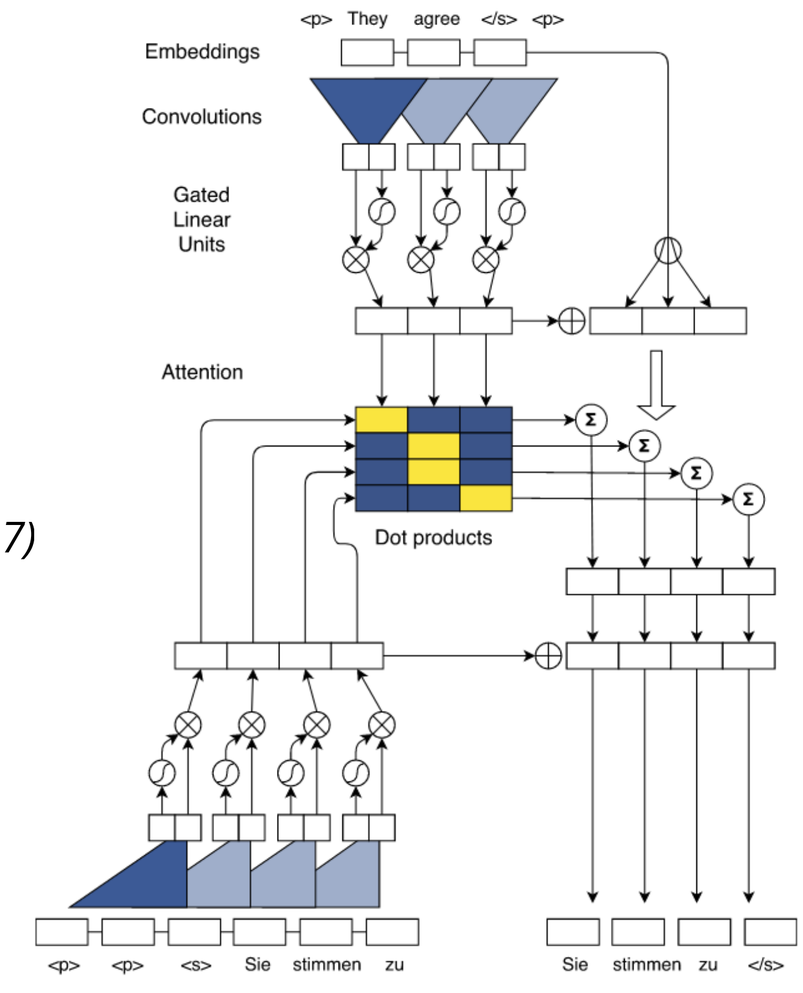
What's next ?

Issues with **recurrent** encoders and decoders:

- Relationships between words do not necessarily follow the linear order
- GPUs can perform parallel computations, but recurrent models need to compute states in order, successively

A convolutional seq2seq model is proposed, allowing for **parallel computation of all hidden states**

Convolutional Sequence to Sequence Learning (Gehring et al, 2017)



What's next ?

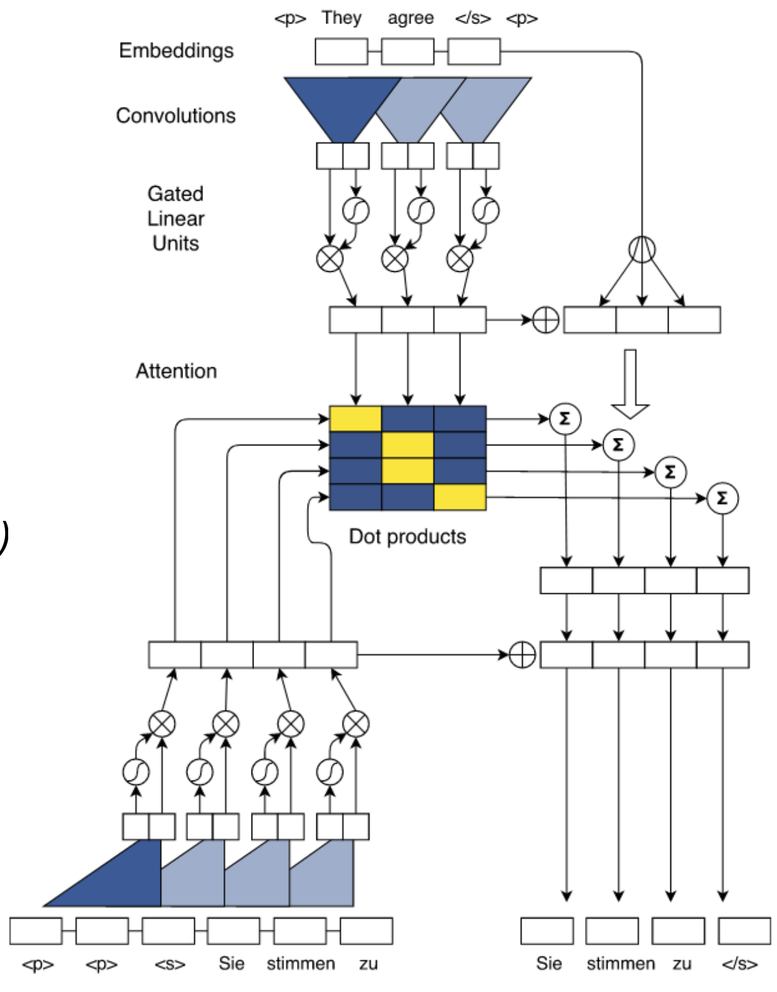
Issues with **recurrent** encoders and decoders:

- Relationships between words do not necessarily follow the linear order
- GPUs can perform parallel computations, but recurrent models need to compute states in order, successively

A convolutional seq2seq model is proposed, allowing for **parallel computation of all hidden states**

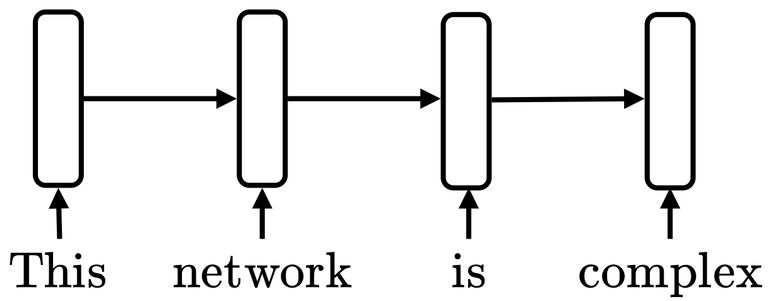
Convolutional Sequence to Sequence Learning (Gehring et al, 2017)

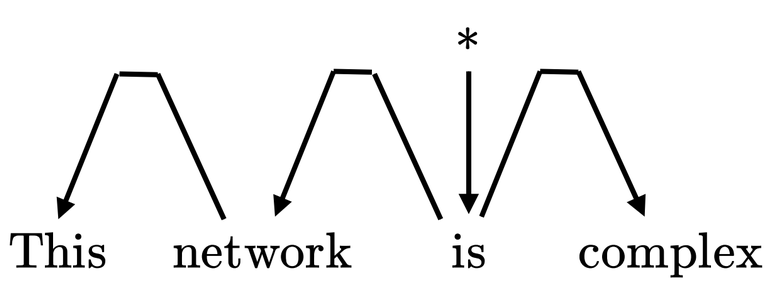
- Still, issues with long-range dependencies

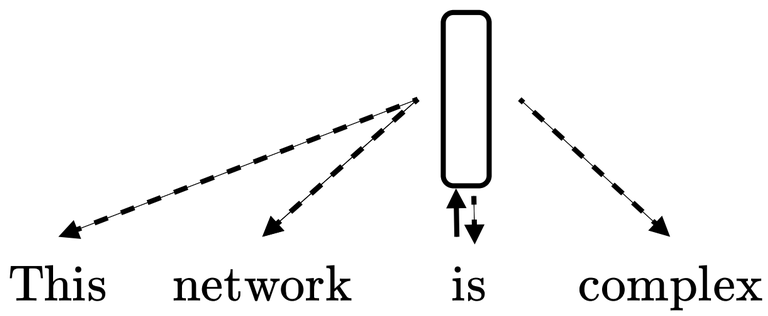


Transformer models

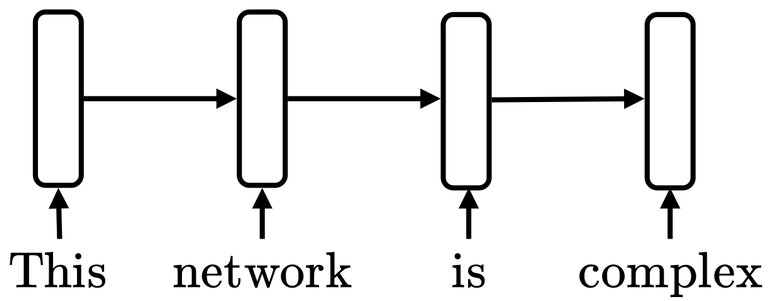
Processing a sequence





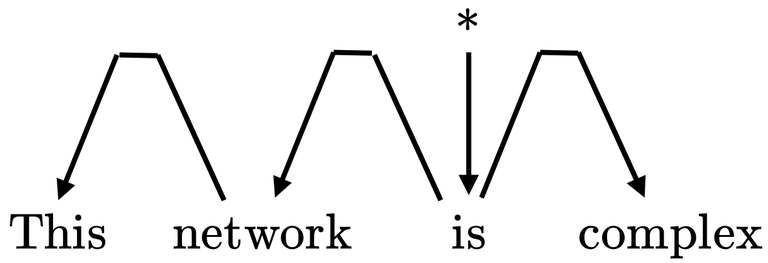


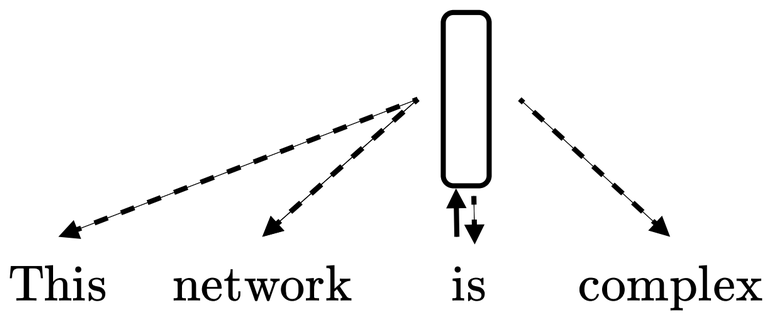
Processing a sequence



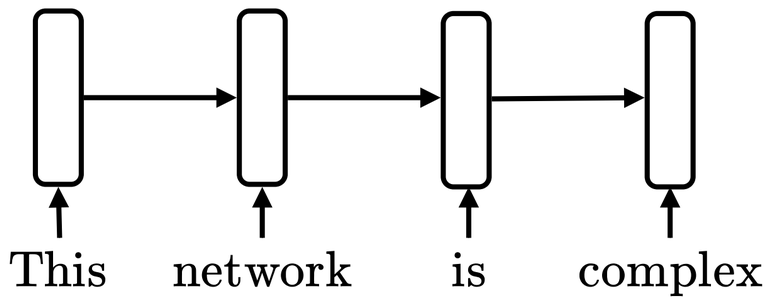
RNN:

- Inputs are fed sequentially
- Updates at each time step



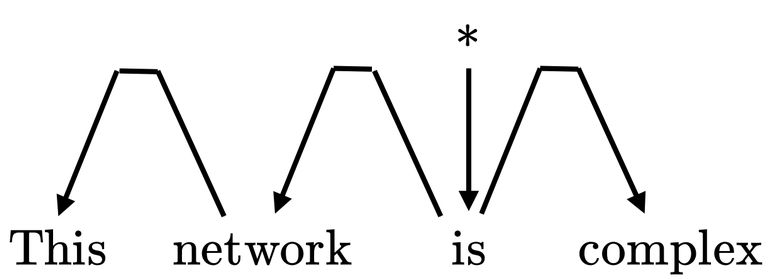


Processing a sequence



RNN:

- Inputs are fed sequentially
- Updates at each time step

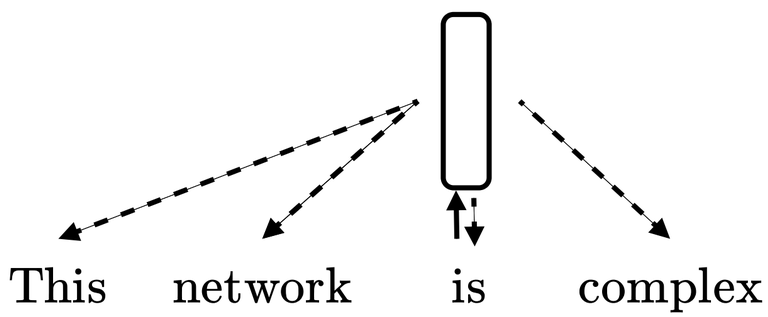


Why not a **GCN** ?

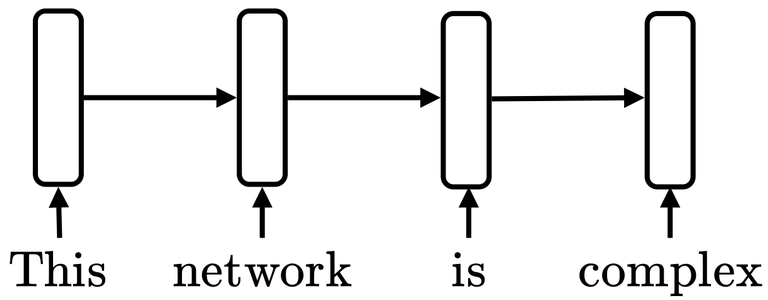
Graph Convolutional Network (Kipf and Welling, 2017)

- Parallelizable
- Long range dependencies

But requires a graph structure !

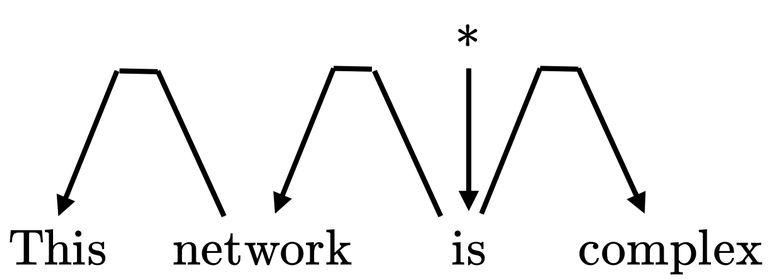


Processing a sequence



RNN:

- Inputs are fed sequentially
- Updates at each time step

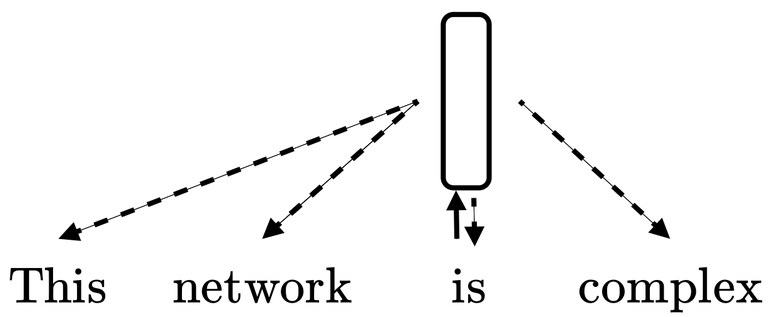


Why not a **GCN** ?

Graph Convolutional Network (Kipf and Welling, 2017)

- Parallelizable
- Long range dependencies

But requires a graph structure !



Idea: combine **Attention**

- Use end-to-end trained attention instead of a given graph structure
- Use multiple attention modules, one for each word

Transformers

Attention is all you Need (Vaswani et al, 2017)

$\rightarrow$ based on **self-attention layers**

- All words in the input sequence are accessible in the same manner
- Can be applied to obtain an output sequence of the same length
- No issue with parallelization

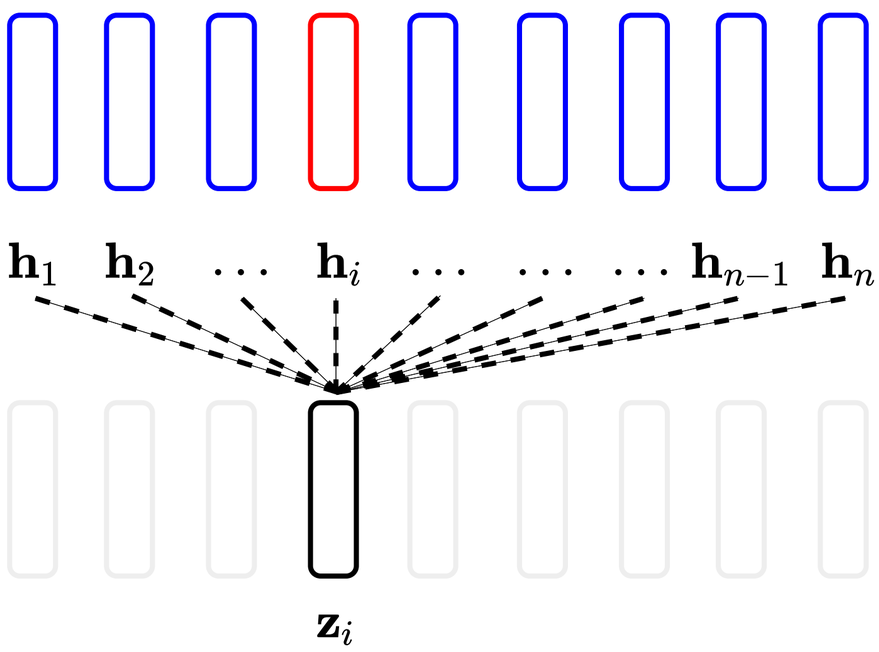
Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

$\rightarrow$ **Transformers are encoder-decoder models where both encoder and decoders are also build with attention**

- However, we will see that they are harder to optimize thand recurrent networks...

Transformers : self-Attention layer

$\rightarrow$ In self-attention, the **queries**, **keys**, and **values** are drawn from the same source:



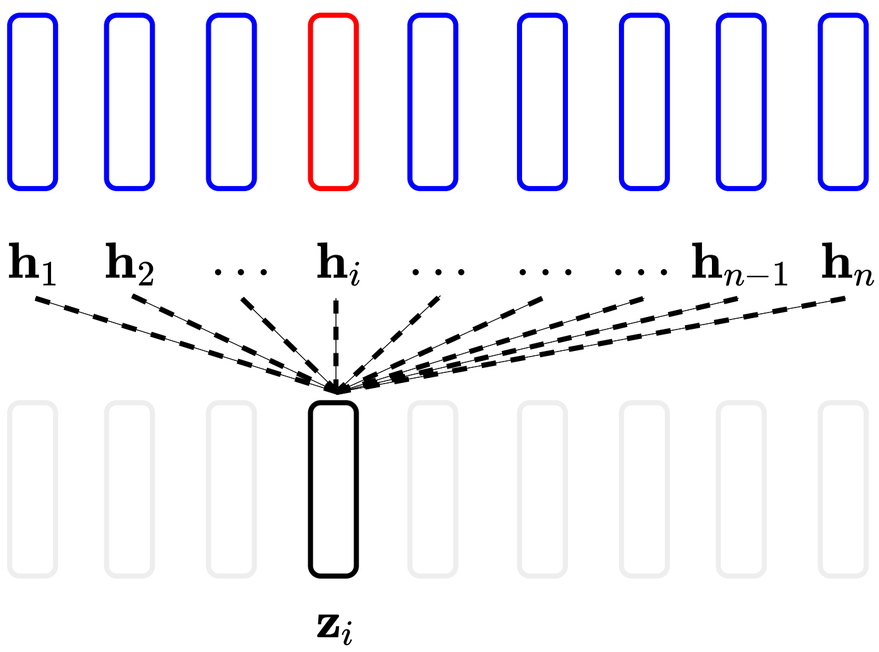
$$\mathbf{e}_j^{(i)} = f(\mathbf{h}_i, \mathbf{h}_j)$$

$$\mathbf{a}_j^{(i)} = \text{softmax}(\mathbf{e}^{(i)})_j$$

$$\mathbf{z}_i = \sum_{j=1}^{n} a_j^{(i)} \mathbf{h}_j$$

Transformers : self-Attention layer

$\rightarrow$ In self-attention, the **queries**, **keys**, and **values** are drawn from the same source:



$$\mathbf{e}_j^{(i)} = f(\mathbf{h}_i, \mathbf{h}_j)$$

$$\mathbf{a}_j^{(i)} = \mathrm{softmax}(\mathbf{e}^{(i)})_j$$

$$\mathbf{z}_i = \sum_{j=1}^{n} a_j^{(i)} \mathbf{h}_j$$

In practice, the **queries**, **keys**, and **values** are computed as linear projections of the same hidden representations:

$$\mathbf{K} = \mathbf{W}_K \mathbf{H}$$
$$\mathbf{Q} = \mathbf{W}_Q \mathbf{H}$$
$$\mathbf{V} = \mathbf{W}_V \mathbf{H}$$

where $\mathbf{W}_K, \mathbf{W}_K \in \mathbb{R}^{d \times d_k}$ project to the same dimension d_k, and $\mathbf{W}_V \in \mathbb{R}^{d \times d_o}$ are parameters of the layer

Transformers : self-Attention layer

When choosing the **scalar product** as similarity function, and adding a **scaling factor** to counteract values getting too large:

$$\mathbf{Z} = Attention(\mathbf{K}, \mathbf{Q}, \mathbf{V}) = softmax\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

- $\mathbf{Q}\mathbf{K}^T \in \mathbb{R}^{n \times n}$: each row contains the importance rows answering: "which input should I look at" ?
- d_k will control the capacity of the self-attention layer.
- $\mathbf{Z}$ contains context-sensitive embeddings for the whole sequence.

Transformers : self-Attention layer

When choosing the **scalar product** as similarity function, and adding a **scaling factor** to counteract values getting too large:

$$\mathbf{Z} = Attention(\mathbf{K}, \mathbf{Q}, \mathbf{V}) = softmax\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

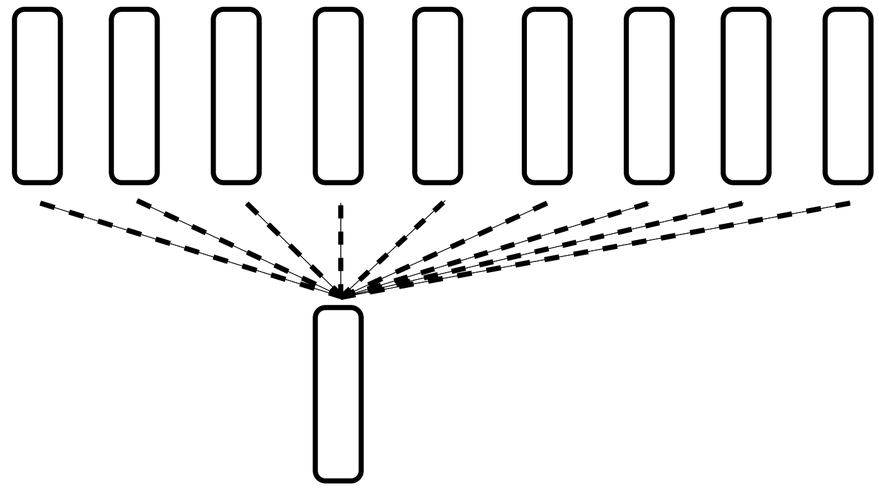
- $\mathbf{Q}\mathbf{K}^T \in \mathbb{R}^{n \times n}$: each row contains the importance rows answering: "which input should I look at" ?
- d_k will control the capacity of the self-attention layer.
- $\mathbf{Z}$ contains context-sensitive embeddings for the whole sequence.

Can we stack self-attention to replace a recurrent layer ? **Not yet !**

- There is **no non-linearities**
- There is **no notion of order** between words
- How can we **not use future words** for autoregressive prediction ?

Transformers : non-linearities

There is **no non-linearities ?**
→ Solution: **add a feed forward network on top of the self-attention output.**



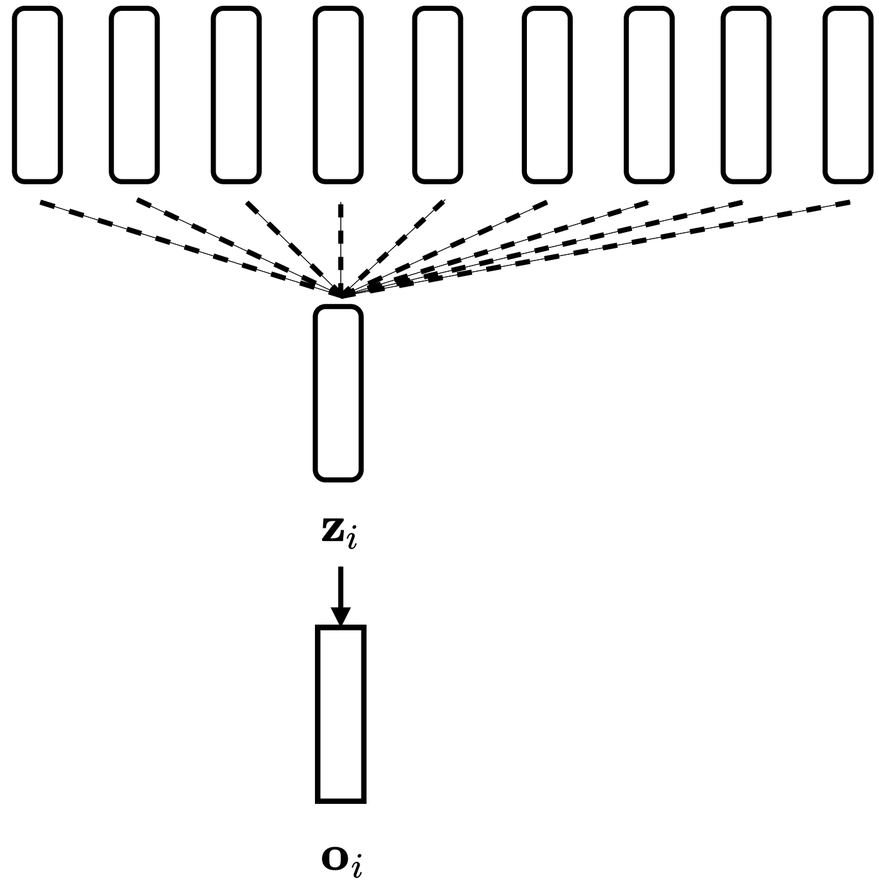
$\mathbf{z}_i$

Transformers : non-linearities

There is **no non-linearities ?**

$\rightarrow$ Solution: **add a feed forward network on top of the self-attention output.**

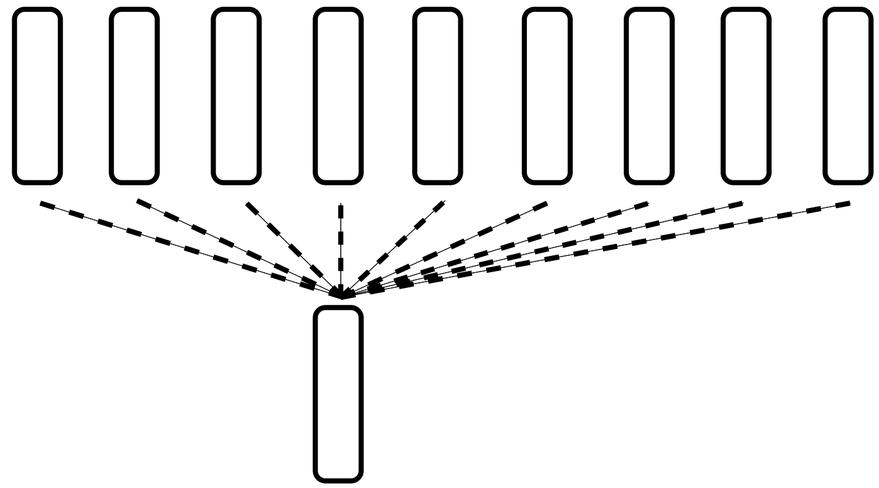
$$\mathbf{o}_i = \mathbf{W}_2 \times \mathrm{ReLU}(\mathbf{W}_1 \times \mathbf{z}_i)$$



Transformers : non-linearities

There is **no non-linearities ?**
$\rightarrow$ Solution: **add a feed forward network on top of the self-attention output.**

$$\mathbf{o}_i = \mathbf{W}_2 \times \mathrm{ReLU}(\mathbf{W}_1 \times \mathbf{z}_i)$$



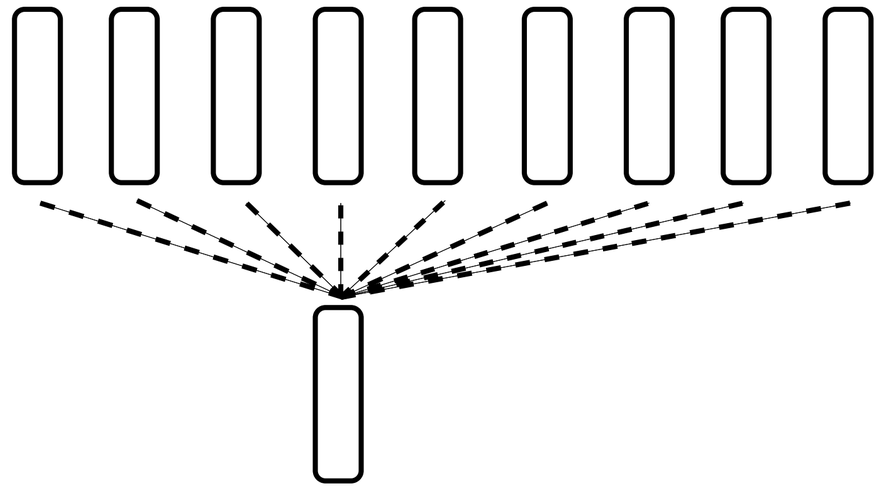
Then,

- **Residual connections**: allows easier training for very deep models by allowing the gradient to 'bypass' complicated operations

Transformers : non-linearities

There is **no non-linearities ?**
$\rightarrow$ Solution: **add a feed forward network on top of the self-attention output.**

$$\mathbf{o}_i = \mathbf{W}_2 \times \mathrm{ReLU}(\mathbf{W}_1 \times \mathbf{z}_i)$$



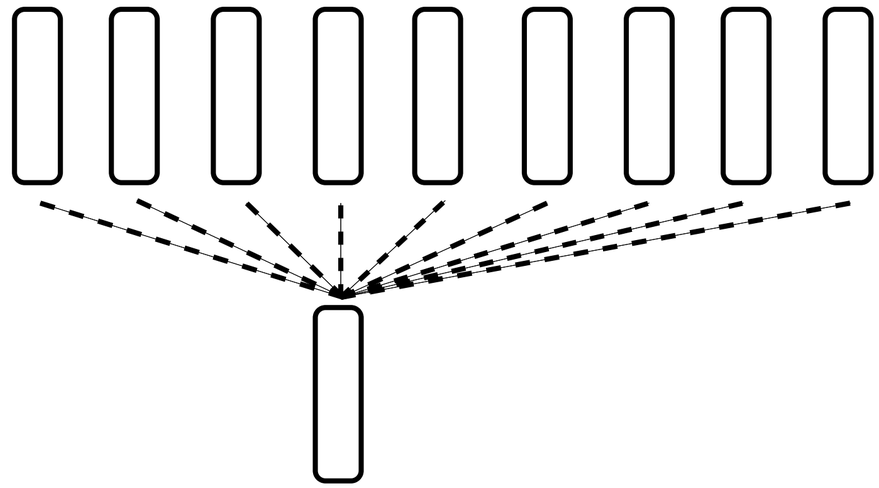
Then,

- **Residual connections**: allows easier training for very deep models by allowing the gradient to 'bypass' complicated operations
- **Layer normalization** cuts down on uninformative variation in hidden vector values by normalizing to unit mean and standard deviation within each layer.

Transformers : non-linearities

There is **no non-linearities ?**
$\rightarrow$ Solution: **add a feed forward network on top of the self-attention output.**

$$\mathbf{o}_i = \mathbf{W}_2 \times \mathrm{ReLU}(\mathbf{W}_1 \times \mathbf{z}_i)$$



Then,

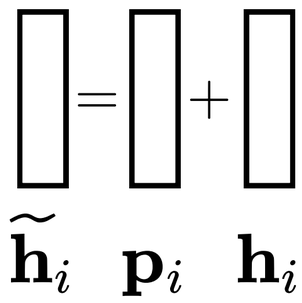
- **Residual connections**: allows easier training for very deep models by allowing the gradient to 'bypass' complicated operations
- **Layer normalization** cuts down on uninformative variation in hidden vector values by normalizing to unit mean and standard deviation within each layer.

These will be applied to this feed-forward network but also accross the whole layer.

Transformers : positional embeddings

There is **no notion of order** between words

$\rightarrow$ Solution: **use a positional embedding to add to representations**



Possibilites are:

Transformers : positional embeddings

There is **no notion of order** between words

$\rightarrow$ Solution: **use a positional embedding to add to representations**

$$\underset{\sim}{\mathbf{h}_i} = \mathbf{h}_i + \mathbf{p}_i$$

Possibilites are:

- $\mathbf{p}_i$ can be represented by learnable parameters, which are randomly initialized and trained end-to-end.

Transformers : positional embeddings

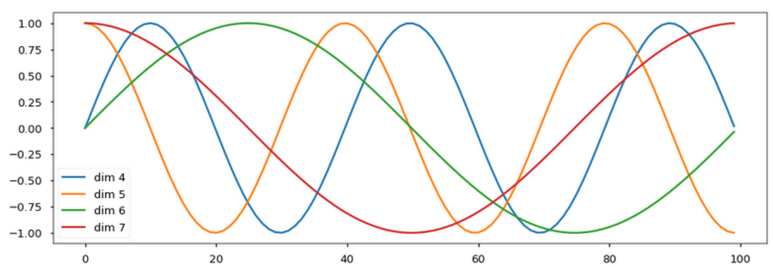
There is **no notion of order** between words

$\rightarrow$ Solution: **use a positional embedding to add to representations**

$$\tilde{\mathbf{h}}_i = \mathbf{h}_i + \mathbf{p}_i$$

Possibilites are:

- $\mathbf{p}_i$ can be represented by learnable parameters, which are randomly initialized and trained end-to-end.
- Fixed embeddings: use cos and sin of different frequencies as positional features.



From "The Annotated Transformer", Havard NLP

Transformers : positional embeddings

There is **no notion of order** between words

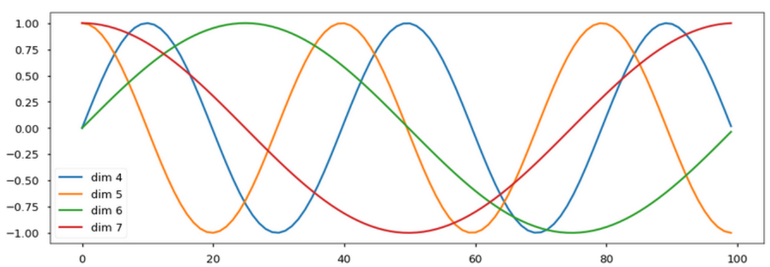
$\rightarrow$ Solution: **use a positional embedding to add to representations**

$$\tilde{\mathbf{h}}_i = \mathbf{h}_i + \mathbf{p}_i$$

Possibilites are:

- $\mathbf{p}_i$ can be represented by learnable parameters, which are randomly initialized and trained end-to-end.
- Fixed embeddings: use cos and sin of different frequencies as positional features.

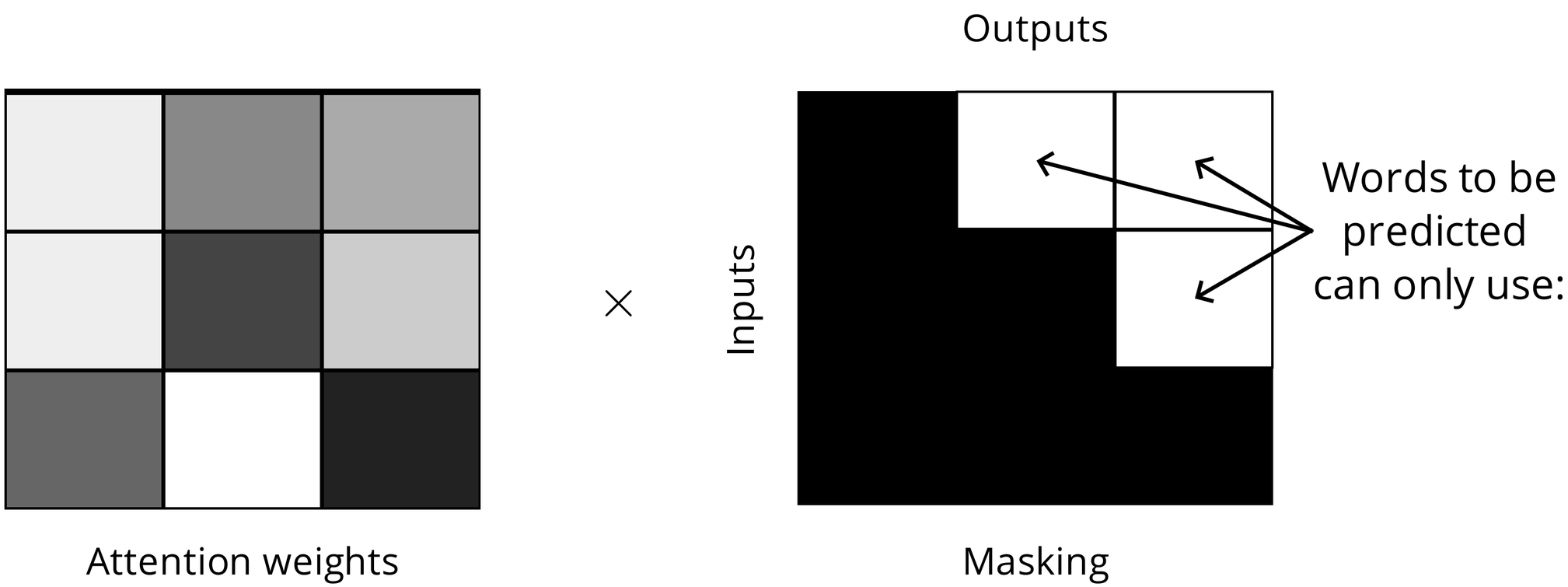
- However, both solutions can not extrapolate to longer sequences !



From "The Annotated Transformer", Havard NLP

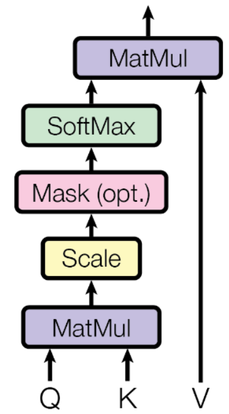
Transformers : masking the future

- How can we enforce **not using future words** for language modeling ?
  $\rightarrow$ Change the set of keys and queries at each time step ? It would be inefficient !
  $\rightarrow$ We **mask out attention to future words** by setting scores to $-\infty$!



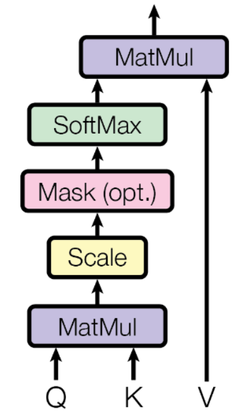
Building a transformer block

Scaled dot-product attention: until now, we have an efficient block that looks like this:



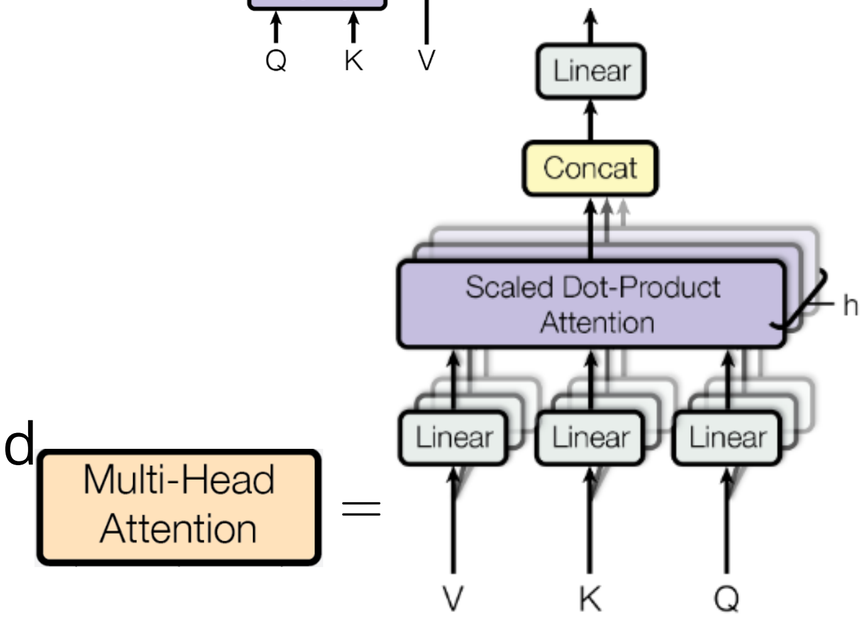
Building a transformer block

Scaled dot-product attention: until now, we have an efficient block that looks like this:



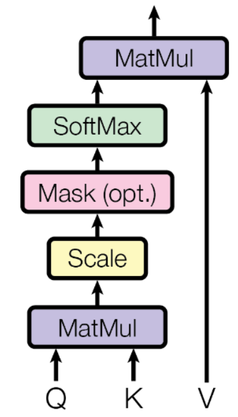
We add **Multi-head Attention**: allowing for many different ways for words to interact together

- Each attention head operates attention independantly, allowing to build different kind of features
- Outputs for differend heads are combined



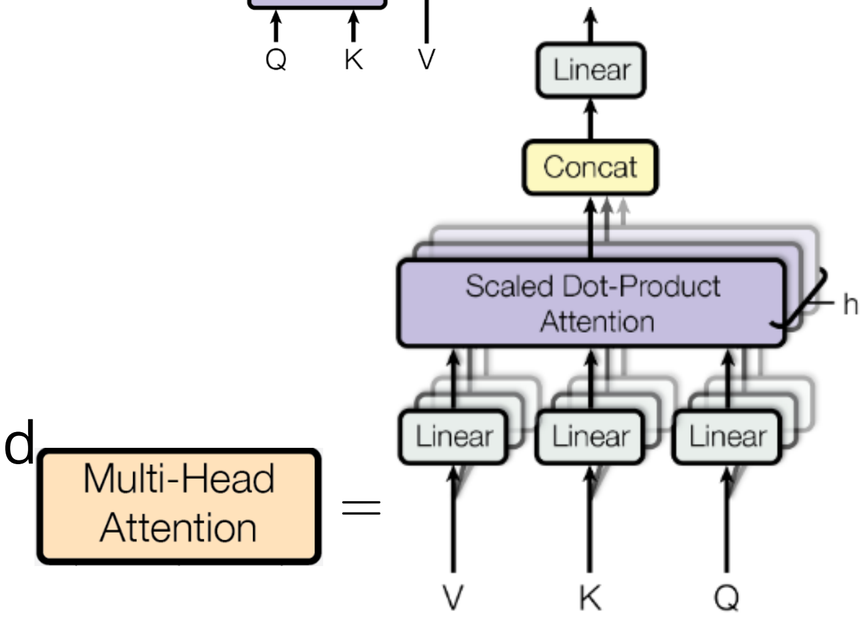
Building a transformer block

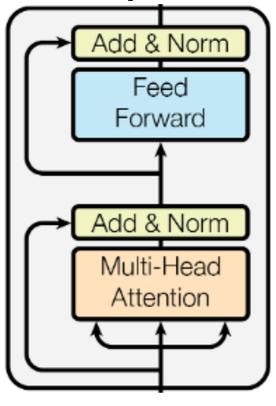
Scaled dot-product attention: until now, we have an efficient block that looks like this:



We add **Multi-head Attention**: allowing for many different ways for words to interact together

- Each attention head operates attention independantly, allowing to build different kind of features
- Outputs for differend heads are combined

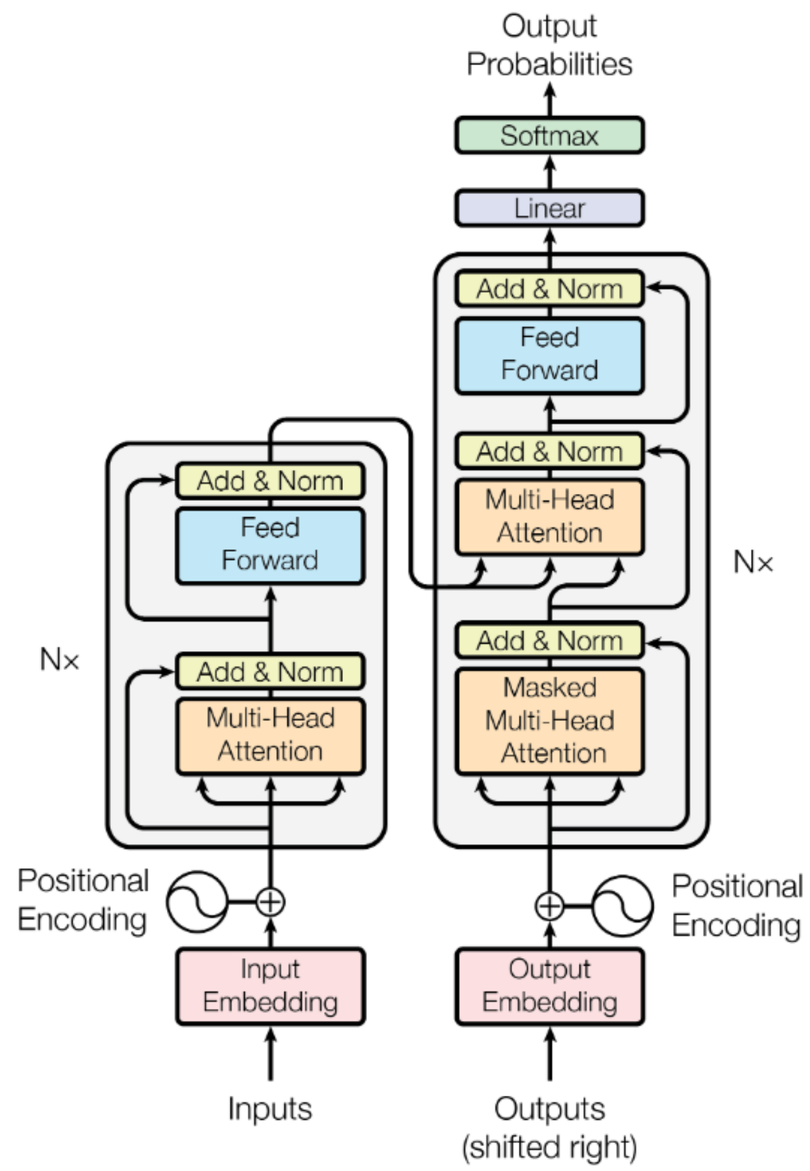




→ With residual connections, layer normalization and a feedforward layer, we finally obtain a **complete transformer block !**

The transformer encoder-decoder

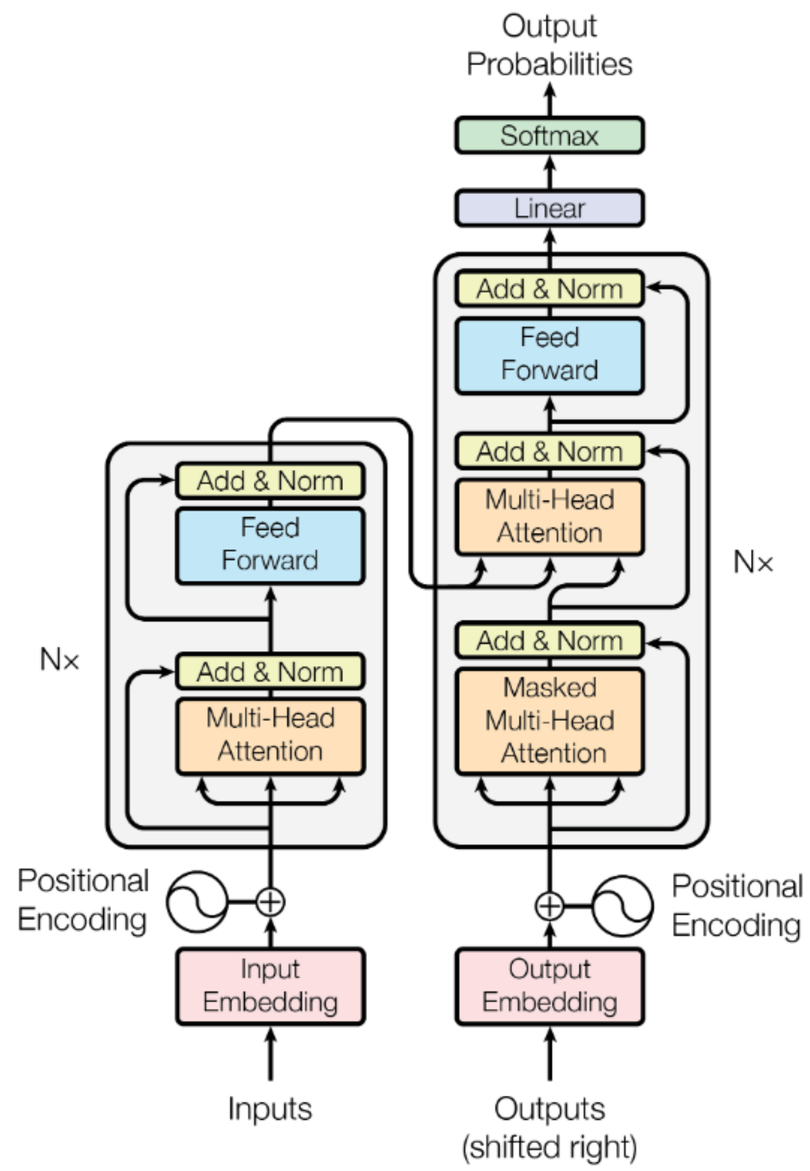
- The input and output embeddings are added the positional encodings



The transformer encoder-decoder

- The input and output embeddings are added the positional encodings

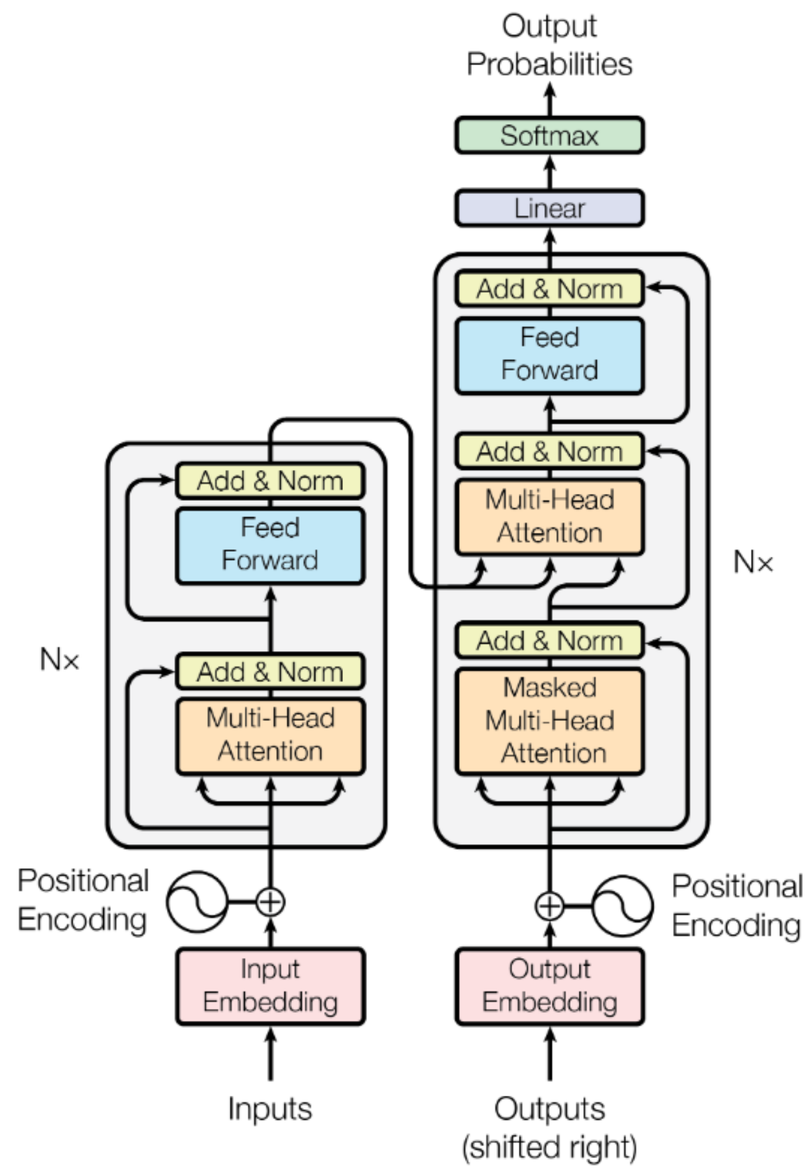
- The encoder is a stack of transformer blocks



The transformer encoder-decoder

- The input and output embeddings are added the positional encodings

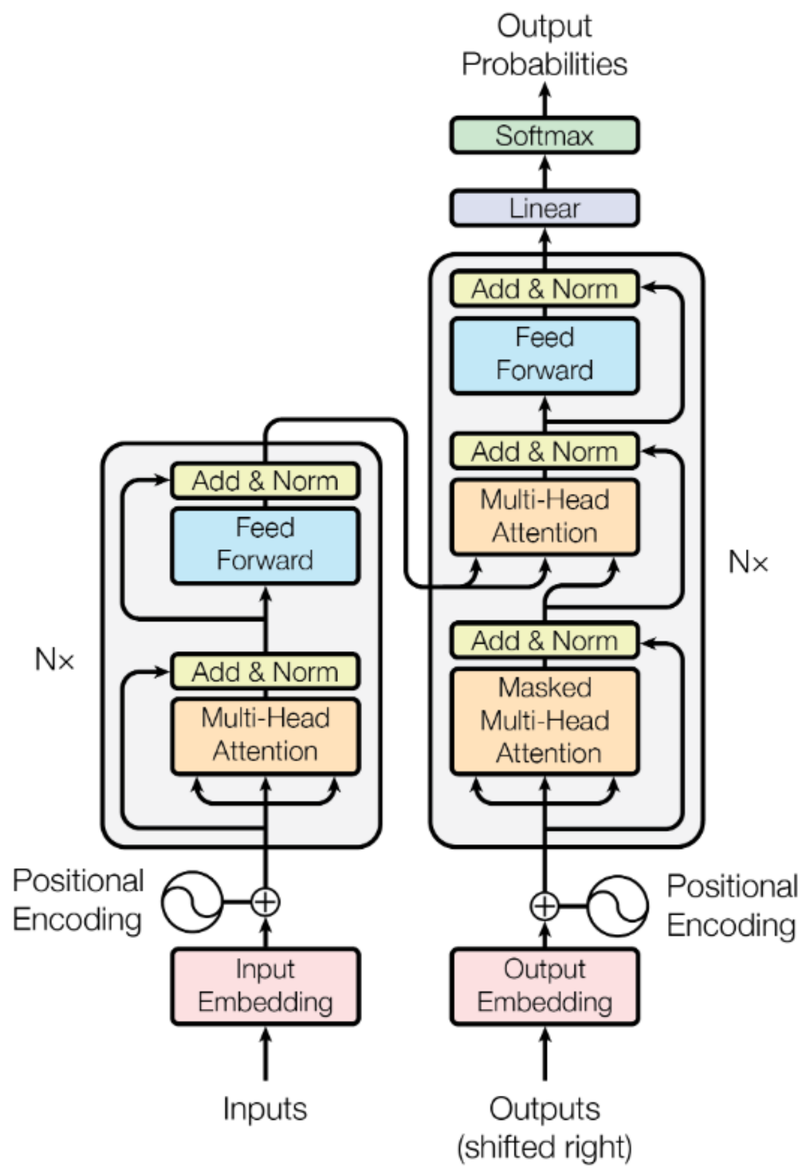
- The encoder is a stack of transformer blocks
- The decoder uses transformer blocks with a first masked multi-head attention to only decode based on previously generated outputs



The transformer encoder-decoder

- The input and output embeddings are added the positional encodings

- The encoder is a stack of transformer blocks
- The decoder uses transformer blocks with a first masked multi-head attention to only decode based on previously generated outputs
- There is **cross attention** between the encoder and decoder, as for the original seq2seq model



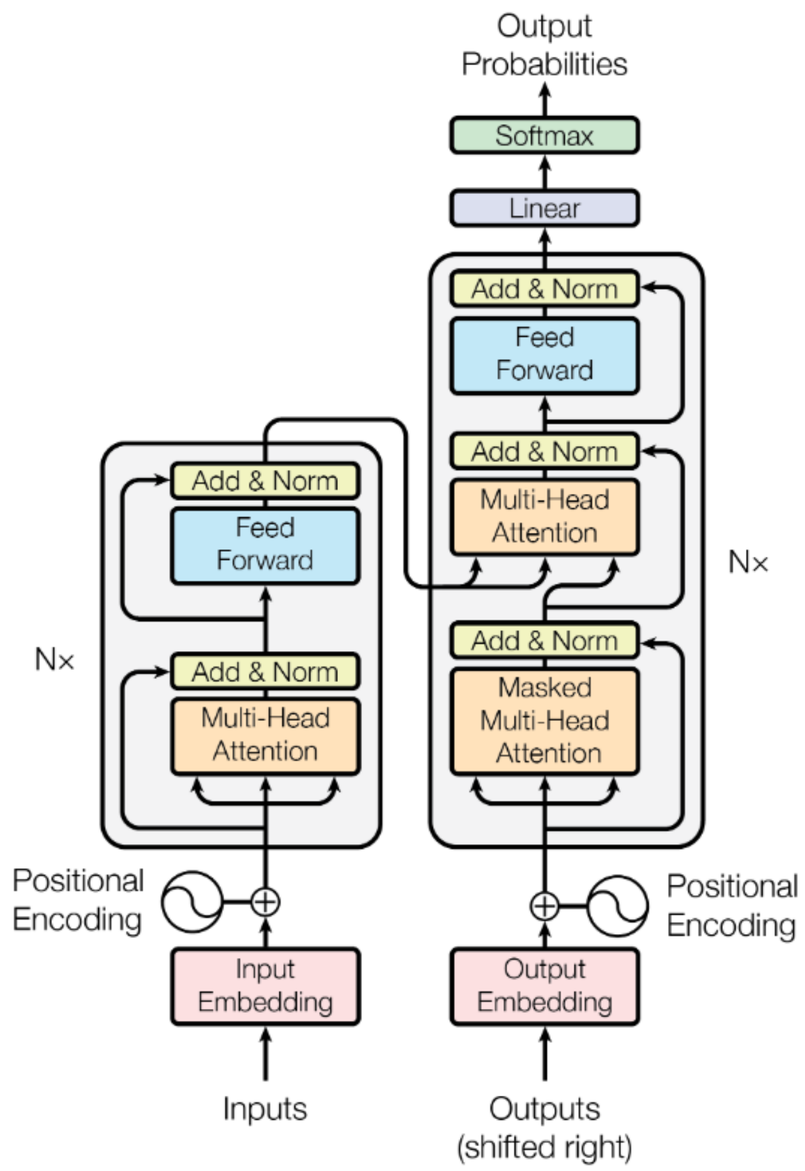
The transformer encoder-decoder

- The input and output embeddings are added the positional encodings

- The encoder is a stack of transformer blocks
- The decoder uses transformer blocks with a first masked multi-head attention to only decode based on previously generated outputs
- There is **cross attention** between the encoder and decoder, as for the original seq2seq model

Attention is all you Need (Vaswani et al, 2017)

- The original paper applied transformers to Machine Translation, obtaining great results !



Transformer model: main advantages

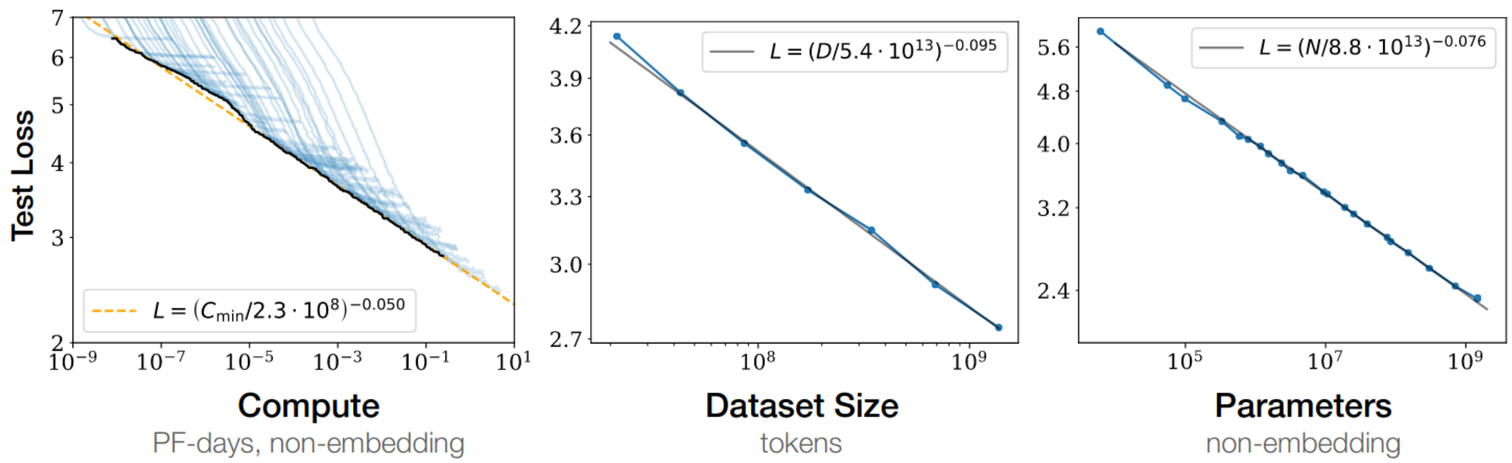
- Easily **parallelizable** > Reccurent networks
- **Constant** maximum path length > Convolutional networks
- Parameter efficient and **length-adaptable** > Dense networks
- Allowed for many models to be trained far longer - with far mode data
  $\rightarrow$ **Scales up** !

Transformer model: main advantages

- Easily **parallelizable** > Reccurent networks
- **Constant** maximum path length > Convolutional networks
- Parameter efficient and **length-adaptable** > Dense networks
- Allowed for many models to be trained far longer - with far mode data
  $\rightarrow$ **Scales up** !

Scaling Laws for Neural Language Models *(Kaplan et al, 2020)*

- For example, language modeling performance improves smoothly as we increase model size, training data, and compute resources.
- **Power-law relationship** observed over multiple orders of magnitude



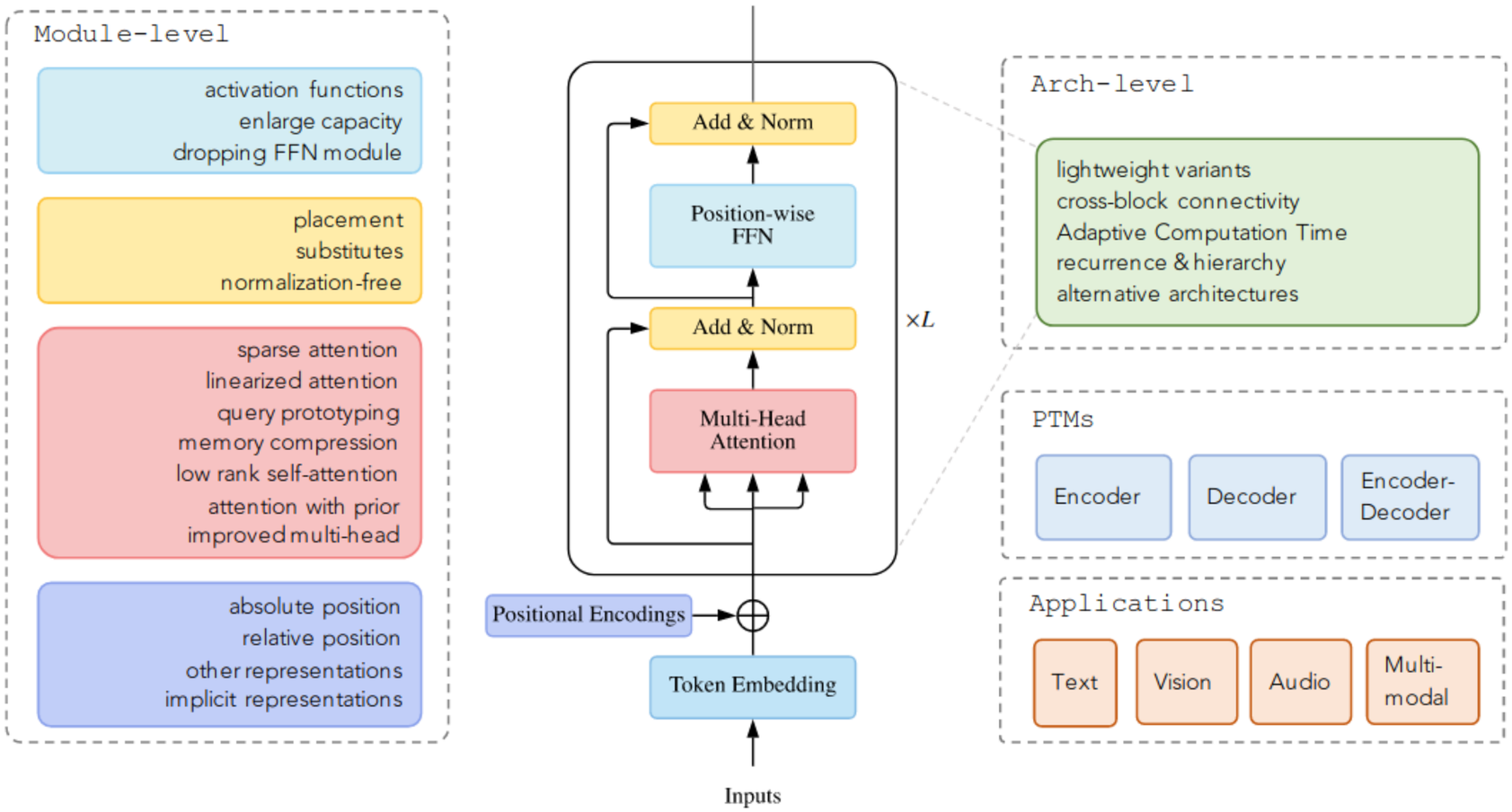
Transformer model: main disadvantages

- No inductive bias ! As such, transformers are prone to overfitting on **small-scale data**
- On small-scale data, the computational bottleneck of the transformer is the feedworward layers - on large-scale data, it is the self-attention
  $\rightarrow$ Both these points show that transformers are not **frugal** !
- By design, there needs to be a maximal sequence length
- Autoregressive generation at inference time is very slow

Further research (that I've been exposed to) seems to focus on:

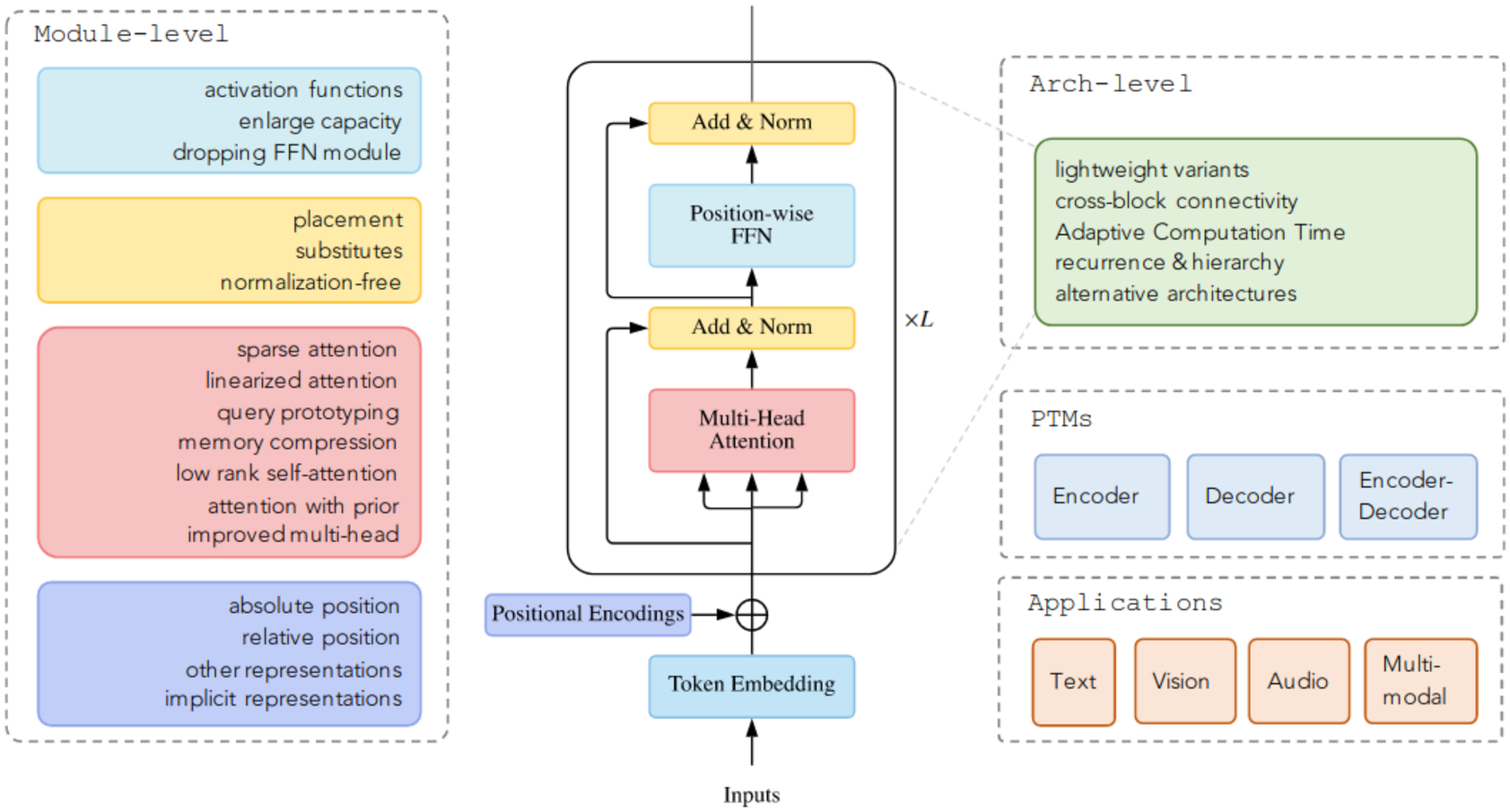
- **Reducing the complexity** of the self-attention layer (which is in $O(n^2)$)
- Any other way to **accelerate** self-attention or transformer training
- Separately of computation time, improving **performance on very long sequences** (>1000 time steps).

Transformers: so many of them



A Survey of Transformers (Lin et al, 2021)

Transformers: so many of them



A Survey of Transformers *(Lin et al, 2021)*

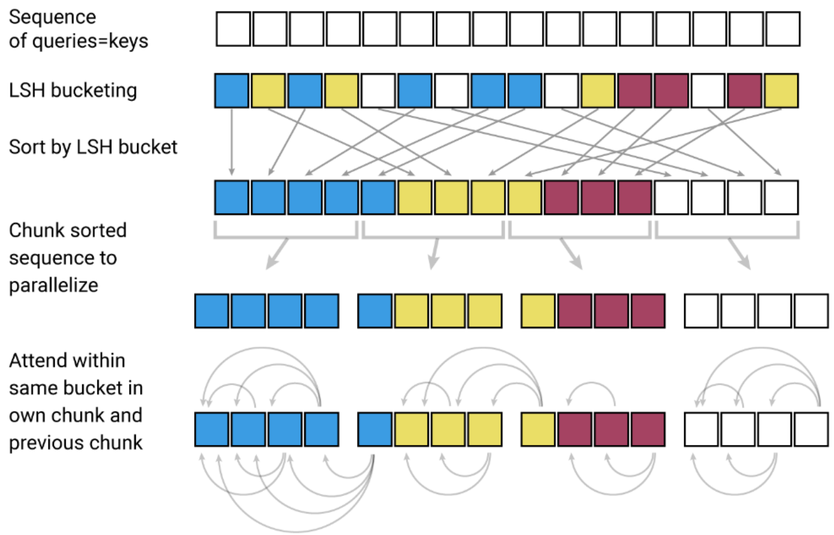
But is that all in vain ? In a study (focusing on NLP): "Surprisingly, we find that most modifications do not meaningfully improve performance."

Do Transformer Modifications Transfer Across Implementations and Applications? *(Narang et al, 2021)*

Transformers: Reducing complexity

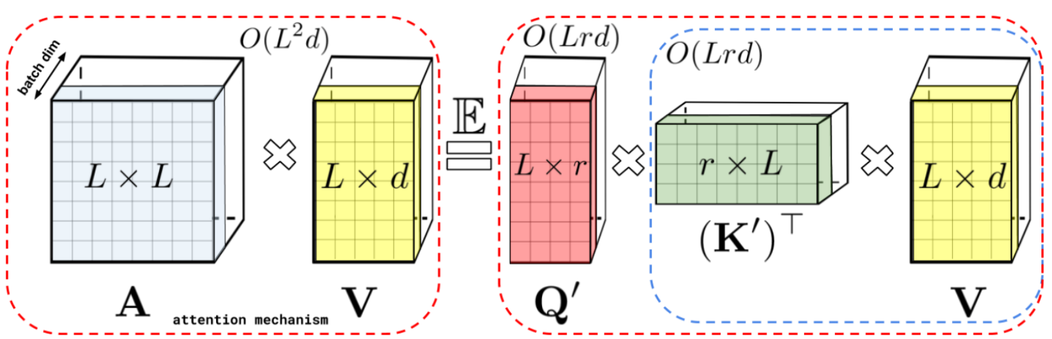
Reformer: the Efficient Transformer (Kitaev et al, 2020)

Idea: use locality-sensitive-hashing (LSH) to reduce the complexity of attending over long sequences, by putting states with a similar hash into the same chunk.



Rethinking Attention with Transformers (Choromanski et al, 2021)

Idea: approximate the softmax using random features maps for linear time and space complexities.



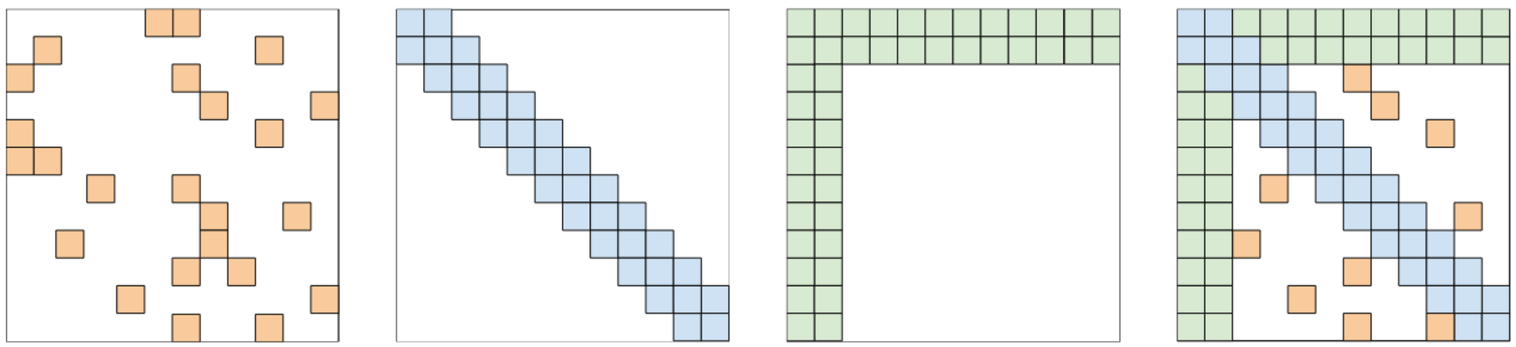
Transformers: Reducing complexity

Nyströmformer: A Nyström-based Algorithm for Approximating Self-Attention (Xiong et al, 2021)

Idea: approximate standard self-attention with $O(n)$ complexity - the algorithm makes use of landmark (or Nyström) points to reconstruct the softmax matrix in self-attention.

Big Bird: Transformers for Longer Sequences (Zaheer et al, 2021)

Idea: replace all-pairs interactions with other kind of interactions, like local windows, looking at everything, and random interactions.



(a) Random attention (b) Window attention (c) Global Attention (d) BIGBIRD

Contextual Representations

Application to Word Representations

Until now: with **pre-trained word representations** (Word2Vec, Gl0ve, ...), NLP saw a lot of progress

- At first, them being pre-trained allowed to **make supervised neural network the state-of-the-art** on numerous tasks:

Application to Word Representations

Until now: with **pre-trained word representations** (Word2Vec, Gl0ve, ...), NLP saw a lot of progress

- At first, them being pre-trained allowed to **make supervised neural network the state-of-the-art** on numerous tasks:
  - Fix them (no updates) and train the rest of the model $\rightarrow$ Knowledge of these words and their relationship is transfered, but not tied to the task !

Application to Word Representations

Until now: with **pre-trained word representations** (Word2Vec, Gl0ve, ...), NLP saw a lot of progress

- At first, them being pre-trained allowed to **make supervised neural network the state-of-the-art** on numerous tasks:
  - Fix them (no updates) and train the rest of the model $\rightarrow$ Knowledge of these words and their relationship is transfered, but not tied to the task !
  - Use them as initial representations and train them with the model

    $\rightarrow$ Knowledge is transfered and tied to the task

Application to Word Representations

Until now: with **pre-trained word representations** (Word2Vec, Gl0ve, ...), NLP saw a lot of progress

- At first, them being pre-trained allowed to **make supervised neural network the state-of-the-art** on numerous tasks:
  - Fix them (no updates) and train the rest of the model $\rightarrow$ Knowledge of these words and their relationship is transfered, but not tied to the task !
  - Use them as initial representations and train them with the model

    $\rightarrow$ Knowledge is transfered and tied to the task

Application to Word Representations

Until now: with **pre-trained word representations** (Word2Vec, Gl0ve, ...), NLP saw a lot of progress

- At first, them being pre-trained allowed to **make supervised neural network the state-of-the-art** on numerous tasks:
  - Fix them (no updates) and train the rest of the model $\rightarrow$ Knowledge of these words and their relationship is transfered, but not tied to the task !
  - Use them as initial representations and train them with the model

    $\rightarrow$ Knowledge is transfered and tied to the task

However, they have two main issues:

- They do not distinguish between different senses of a word - **same token, different word type**
- Even the same word can have different meanings, aspects, ... depending on the **context**

Contextual Word Representations: evolution of transfer learning

Main idea:

$\rightarrow$ In a recurrent neural language model, while processing the sequence of words to predict the next one, **the hidden state contains all the contextual information necessary**, at each position !

Contextual Word Representations: evolution of transfer learning

Main idea:

$\rightarrow$ In a recurrent neural language model, while processing the sequence of words to predict the next one, **the hidden state contains all the contextual information necessary**, at each position !

- *Semi-supervised Sequence Learning (Dai et al, 2015)*

  The parameter of an 'unsupervised' language model (or auto-encoder) are used as input for a supervised sequence model.

Contextual Word Representations: evolution of transfer learning

Main idea:

$\rightarrow$ In a recurrent neural language model, while processing the sequence of words to predict the next one, **the hidden state contains all the contextual information necessary**, at each position !

- *Semi-supervised Sequence Learning (Dai et al, 2015)*
  The parameter of an 'unsupervised' language model (or auto-encoder) are used as input for a supervised sequence model.
- *Semi-supervised sequence tagging with bidirectional language models (Peters et al, 2017)*
  Uses both classic embeddings and representations from a Bi-LSTM Language model for sequence tagging.

Contextual Word Representations: evolution of transfer learning

Main idea:

$\rightarrow$ In a recurrent neural language model, while processing the sequence of words to predict the next one, **the hidden state contains all the contextual information necessary**, at each position !

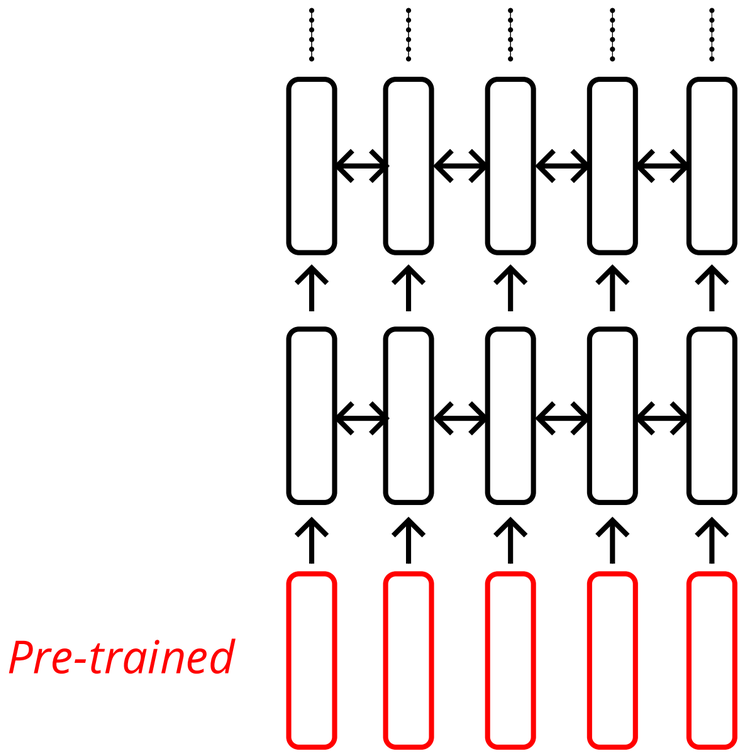
- *Semi-supervised Sequence Learning (Dai et al, 2015)*
  The parameter of an 'unsupervised' language model (or auto-encoder) are used as input for a supervised sequence model.
- *Semi-supervised sequence tagging with bidirectional language models (Peters et al, 2017)*
  Uses both classic embeddings and representations from a Bi-LSTM Language model for sequence tagging.
- *Universal Language Model Fine-tuning for Text Classification (Ruder et al, 2018)*
  Similarly, transfer parameters from an unsupervised language model, but here, the architecture is kept while the final layer is changed.

Contextual Word Representations

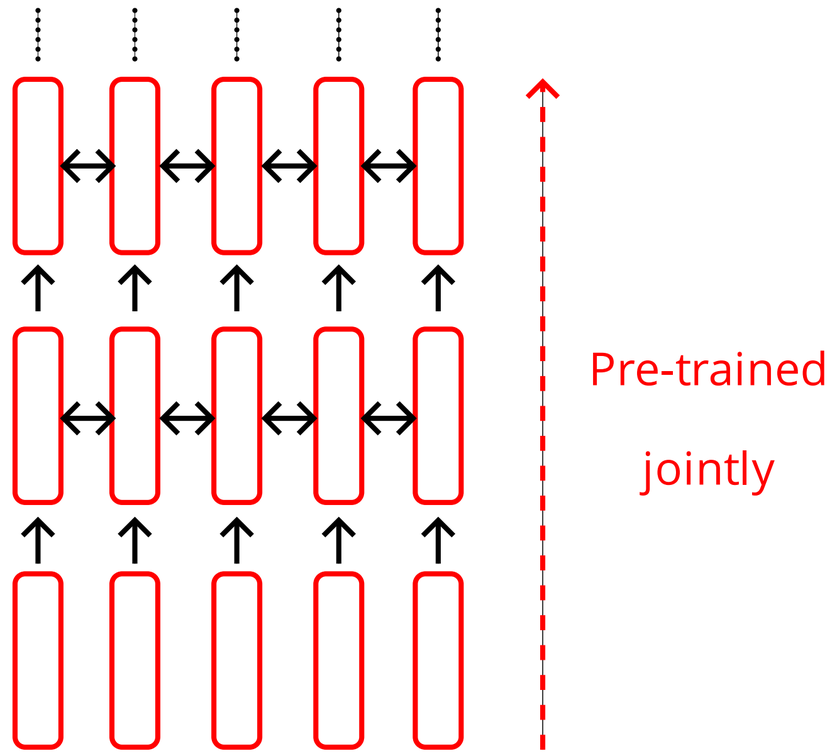
Main idea:

→ In a recurrent neural language model, while processing the sequence of words to predict the next one, **the hidden state contains all the contextual information necessary**, at each position !

Currently:



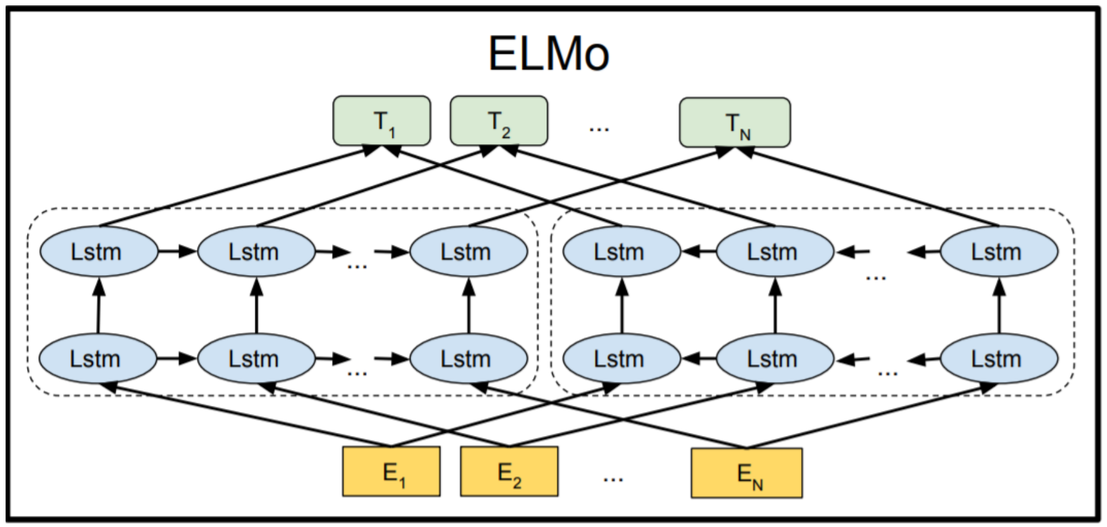
Goal:



Embeddings from Language Models

And then: **ELMO**

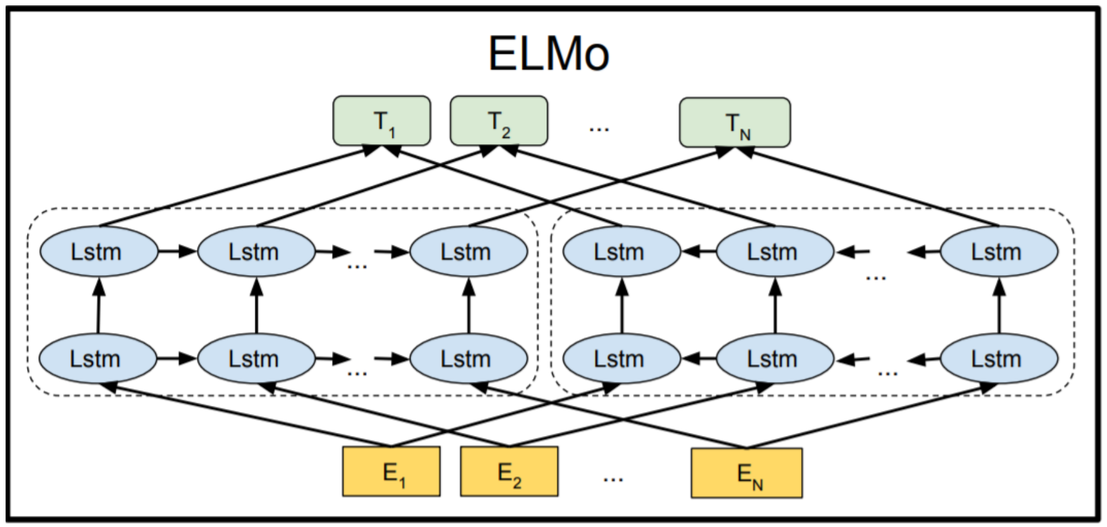
Deep contextualized word representations (Peters et al, 2018)



Embeddings from Language Models

And then: **ELMO**

Deep contextualized word representations (Peters et al, 2018)

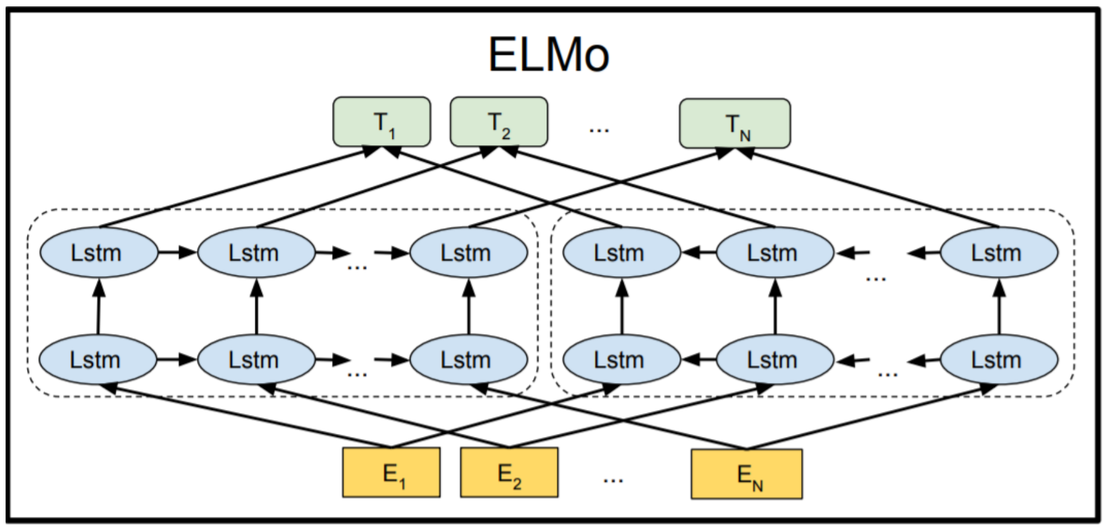


- Independantly train to LSTM Language models (both directions)

Embeddings from Language Models

And then: **ELMO**

Deep contextualized word representations (Peters et al, 2018)



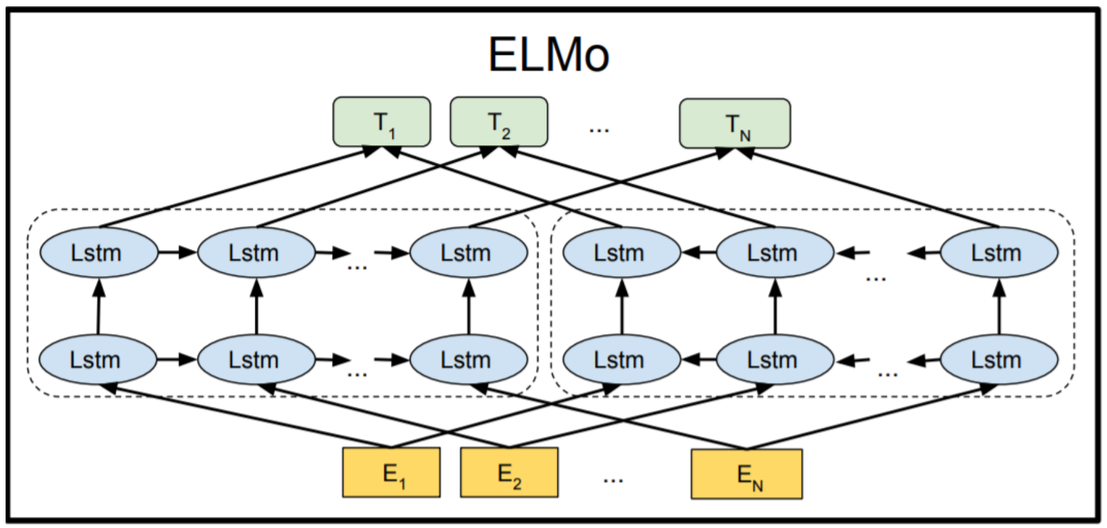
- Independantly train to LSTM Language models (both directions)
- Learns task specific representations which are combinations of LSTM layers in both Models:

$$\mathbf{ELMo}_k^{task} = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}$$

Embeddings from Language Models

And then: **ELMO**

Deep contextualized word representations (Peters et al, 2018)



- Independantly train to LSTM Language models (both directions)
- Learns task specific representations which are combinations of LSTM layers in both Models:

$$\mathbf{ELMo}_k^{task} = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}$$

- Paradigm change: learn build embeddings trough **building a whole sentence** instead of simply predicting a context word

A new trend in NLP

- With **ELMO,** you can train both LMs, freeze the models, and use the representations for a lot of tasks.
- The authors beat (sometimes largely) state of the art systems on several popular tasks by simply using ELMo representations with their own baselines.

$\rightarrow$ It made a huge impact in the field !

A new trend in NLP

- With **ELMO,** you can train both LMs, freeze the models, and use the representations for a lot of tasks.
- The authors beat (sometimes largely) state of the art systems on several popular tasks by simply using ELMo representations with their own baselines.

$\rightarrow$ It made a huge impact in the field !

Contextual word representations produced by very large models **pre-trained** in an **unsupervised manner** via processing huge quantities of data are the new norm for most sequence-based tasks in NLP

For three types of sequential models:

A new trend in NLP

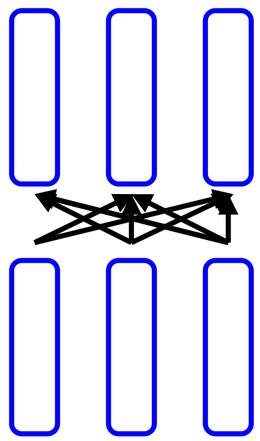
- With **ELMO,** you can train both LMs, freeze the models, and use the representations for a lot of tasks.
- The authors beat (sometimes largely) state of the art systems on several popular tasks by simply using ELMo representations with their own baselines.

→ It made a huge impact in the field !

Contextual word representations produced by very large models **pre-trained** in an **unsupervised manner** via processing huge quantities of data are the new norm for most sequence-based tasks in NLP

For three types of sequential models:

- Encoders



A new trend in NLP

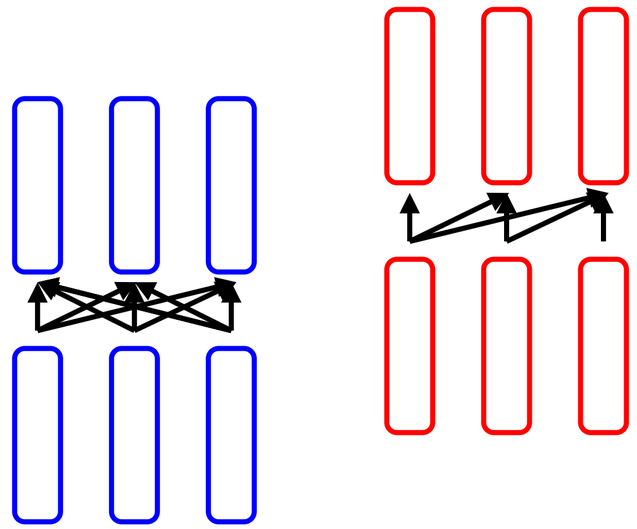
- With **ELMO,** you can train both LMs, freeze the models, and use the representations for a lot of tasks.
- The authors beat (sometimes largely) state of the art systems on several popular tasks by simply using ELMo representations with their own baselines.

$\rightarrow$ It made a huge impact in the field !

Contextual word representations produced by very large models **pre-trained** in an **unsupervised manner** via processing huge quantities of data are the new norm for most sequence-based tasks in NLP

For three types of sequential models:

- Encoders
- Decoders



A new trend in NLP

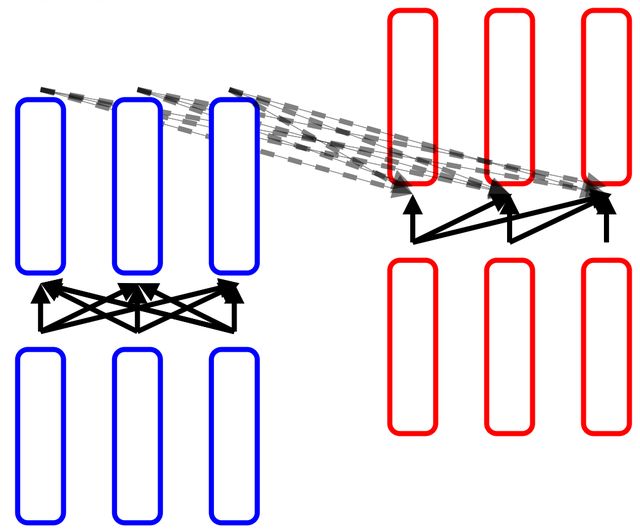
- With **ELMO,** you can train both LMs, freeze the models, and use the representations for a lot of tasks.
- The authors beat (sometimes largely) state of the art systems on several popular tasks by simply using ELMo representations with their own baselines.

$\rightarrow$ It made a huge impact in the field !

Contextual word representations produced by very large models **pre-trained** in an **unsupervised manner** via processing huge quantities of data are the new norm for most sequence-based tasks in NLP

For three types of sequential models:

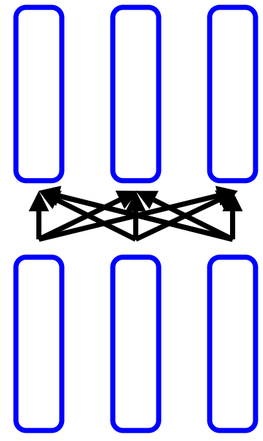
- Encoders
- Decoders
- Encoder-decoders



Pre-training and Fine-tuning transformer models

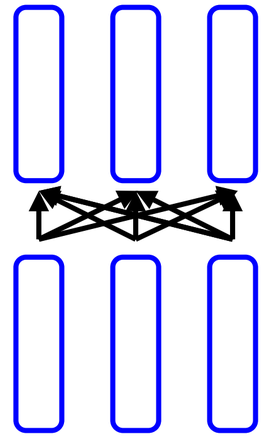
Pre-Training an encoder

- The encoder is **bidirectionnal:** context from both sides
- How to train it to obtain good representations ?



Pre-Training an encoder

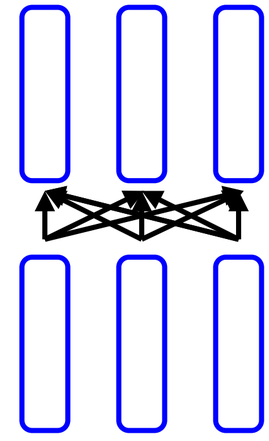
- The encoder is **bidirectionnal:** context from both sides
- How to train it to obtain good representations ?



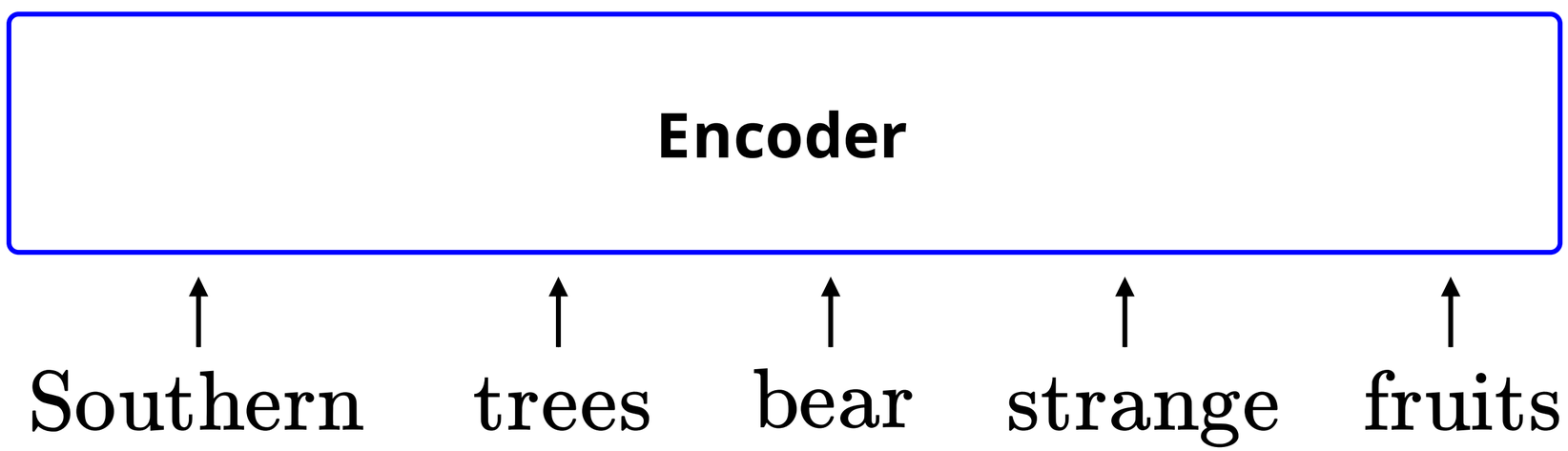
- Remember **Word2ve**c !

Pre-Training an encoder

- The encoder is **bidirectionnal:** context from both sides
- How to train it to obtain good representations ?

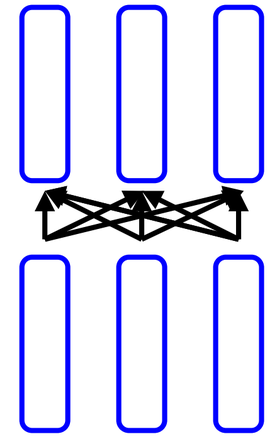


- Remember **Word2ve**c !

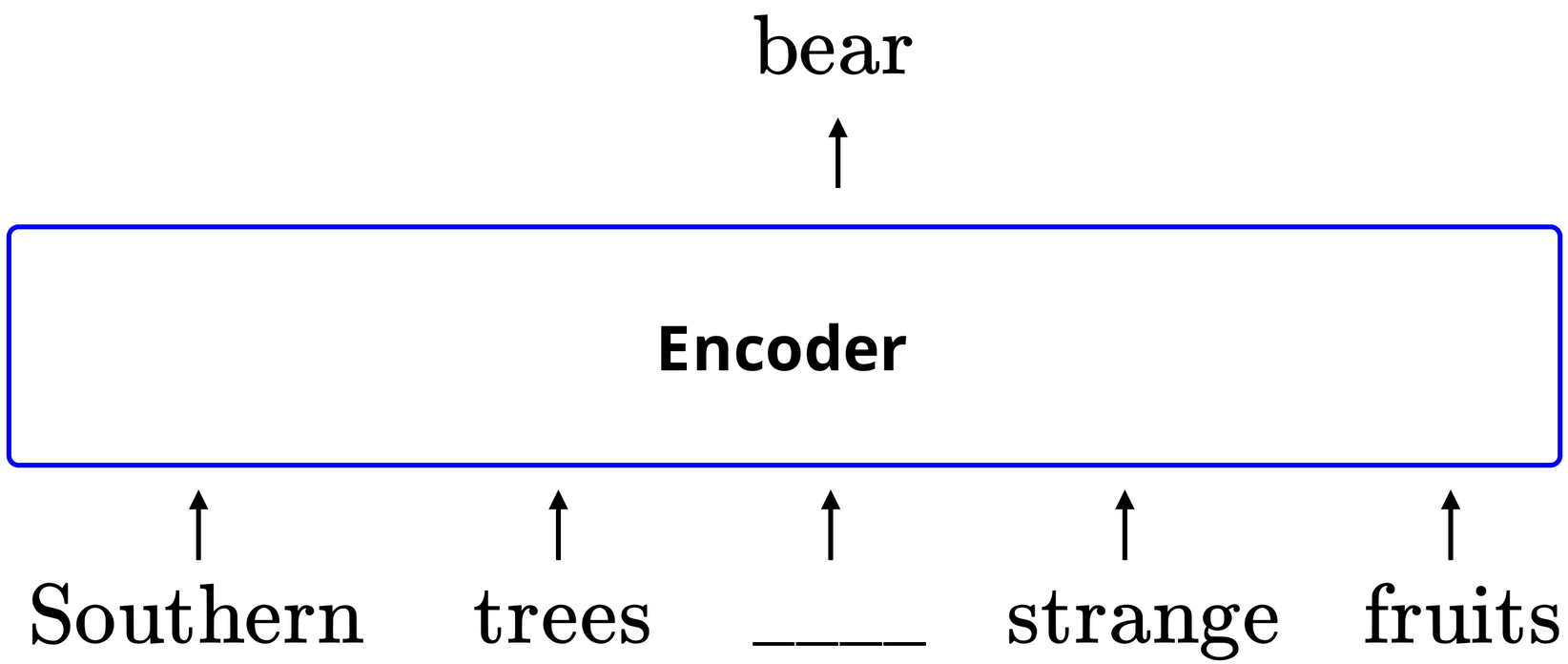


Pre-Training an encoder

- The encoder is **bidirectionnal:** context from both sides
- How to train it to obtain good representations ?



- Remember **Word2ve**c !



Bidirectional Encoder Representations from Transformers

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al, 2018)

Bidirectional Encoder Representations from Transformers

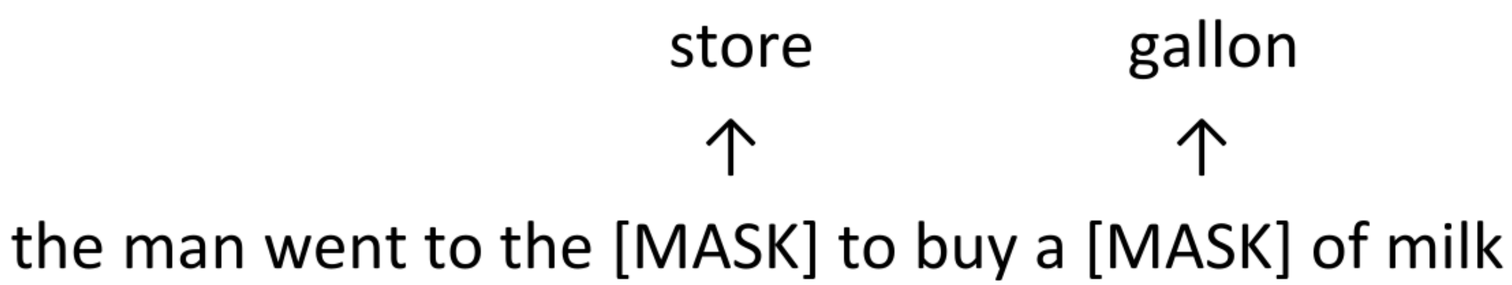
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al, 2018)

- As with ELMo, the goal is to obtain **contextual representations** in an **unsupervised** manner to use for other tasks
  - ELMo concatenates 2 LSTM Language models (Decoders !)
  - BERT uses one **transformer encoder**

Bidirectional Encoder Representations from Transformers

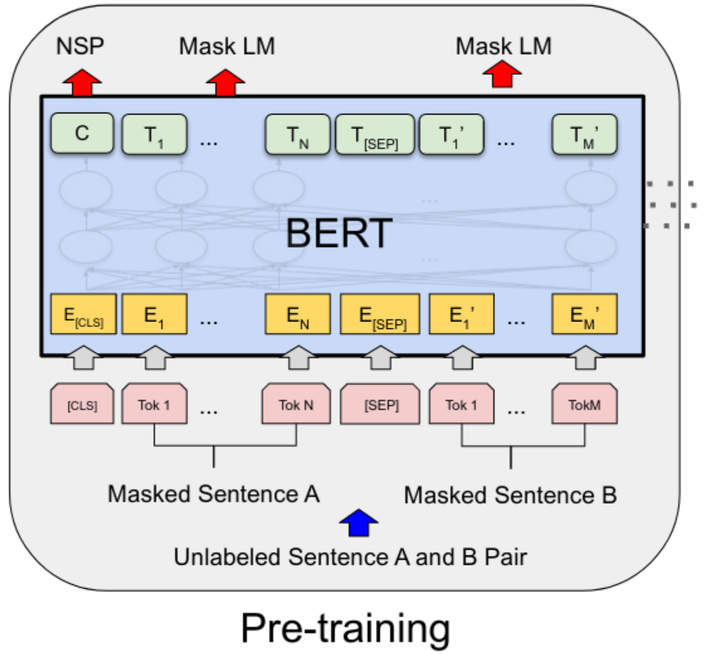
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al, 2018)

- As with ELMo, the goal is to obtain **contextual representations** in an **unsupervised** manner to use for other tasks
  - ELMo concatenates 2 LSTM Language models (Decoders !)
  - BERT uses one **transformer encoder**
- To obain a true bidirectional representation of language, **mask a proportion** of the training input and **predict the missing words** as training objective: this is the **Mask Language Model (MLM)** objective



BERT: Pre-training tasks

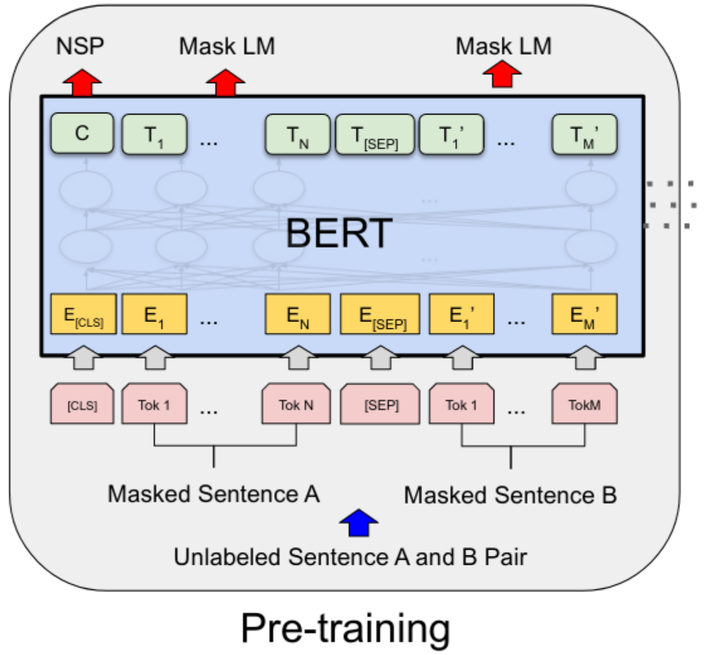
- **Stacked transformer layers** (12 or 24)
- Trained on very large datasets (Wikipedia + BookCorpus)
- Uses Wordpiece with 30.000 tokens



Pre-training

BERT: Pre-training tasks

- **Stacked transformer layers** (12 or 24)
- Trained on very large datasets (Wikipedia + BookCorpus)
- Uses Wordpiece with 30.000 tokens



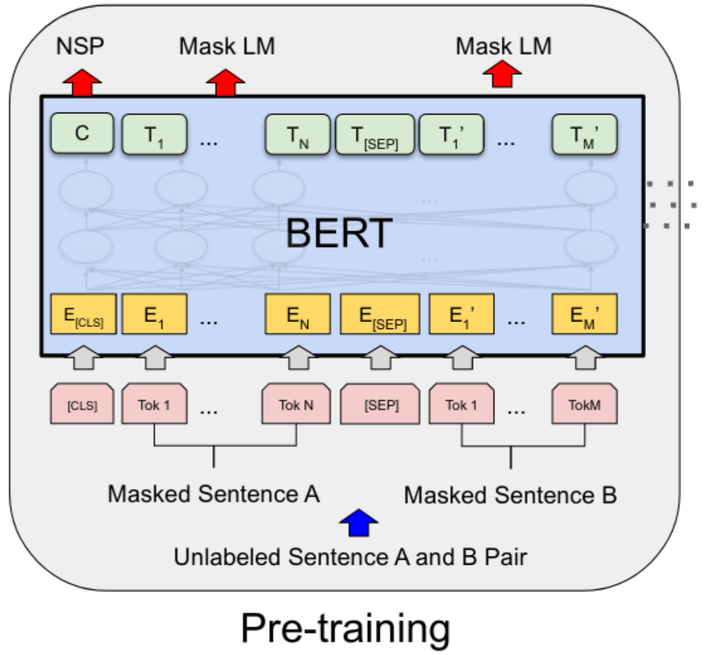
Pre-training

Pre-training tasks:

- **MLM** - mask 15% of tokens and predict them.
- For **robustness**, replace input word with [MASK] 80% of the time - use a random token 10% of the time, or leave it unchanged the last 10%
  $\rightarrow$ The model is also trained to replace incoherent words

BERT: Pre-training tasks

- **Stacked transformer layers** (12 or 24)
- Trained on very large datasets (Wikipedia + BookCorpus)
- Uses Wordpiece with 30.000 tokens



Pre-training

Pre-training tasks:

- **MLM** - mask 15% of tokens and predict them.
- For **robustness**, replace input word with [MASK] 80% of the time - use a random token 10% of the time, or leave it unchanged the last 10%
  $\rightarrow$ The model is also trained to replace incoherent words
- **NSP - next sentence prediction:** does the masked sentence B follows sentence A, or is it a random pairing ?
  $\rightarrow$ Was later shown to be useless !

Fine-tuning an encoder

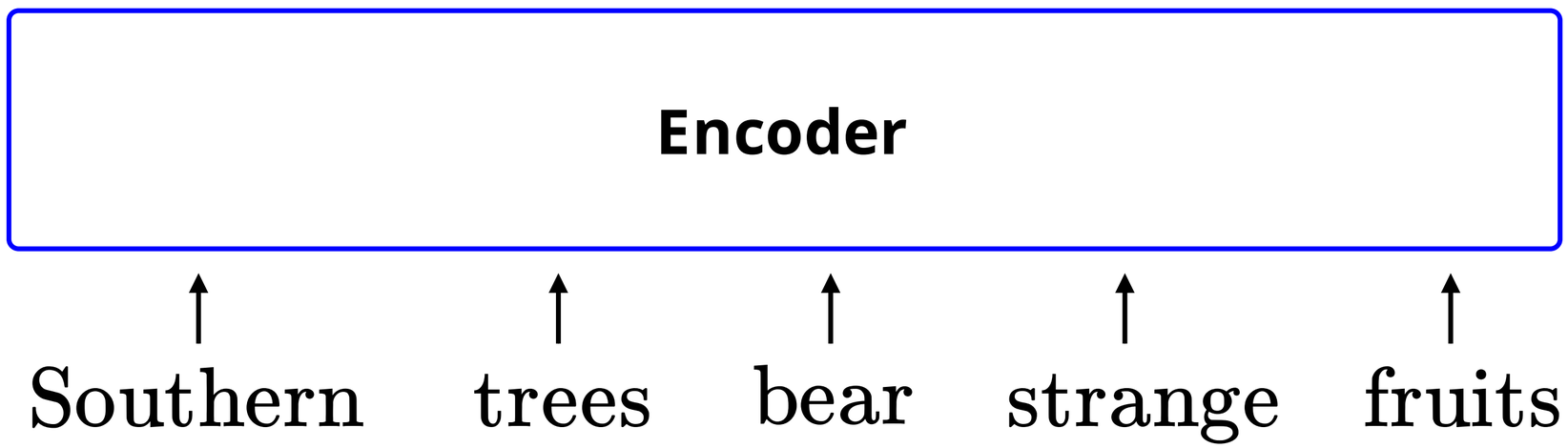
Fine-tuning:

- **Remove the top (prediction) layer corresponding to words** - and replace it with a new layer corresponding to the new task: sequence classification, token classification...

Fine-tuning an encoder

Fine-tuning:

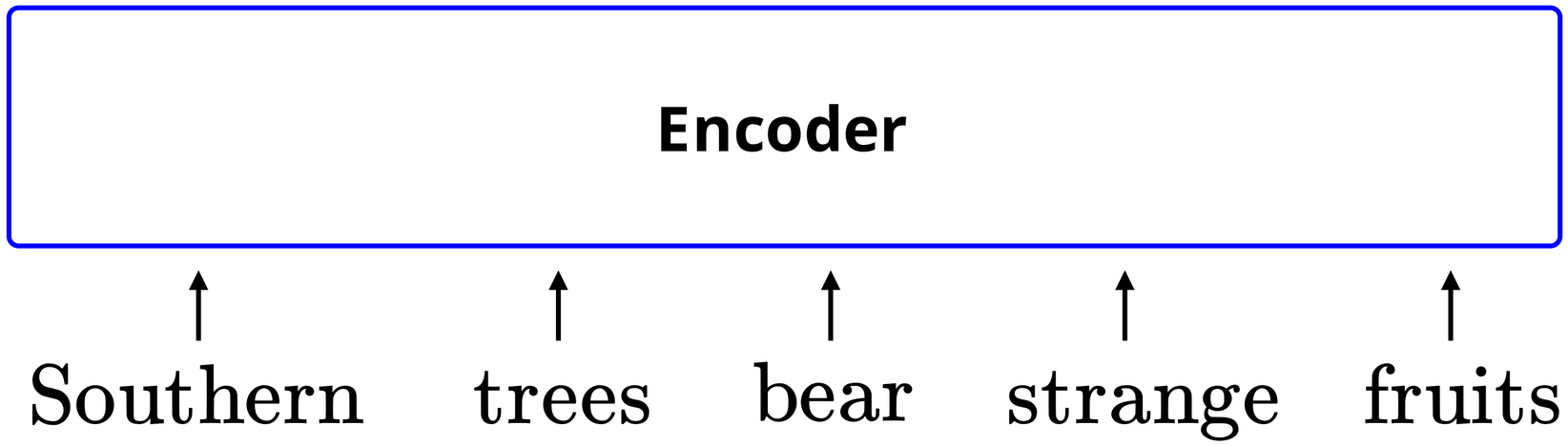
- **Remove the top (prediction) layer corresponding to words** - and replace it with a new layer corresponding to the new task: sequence classification, token classification...



Fine-tuning an encoder

Fine-tuning:

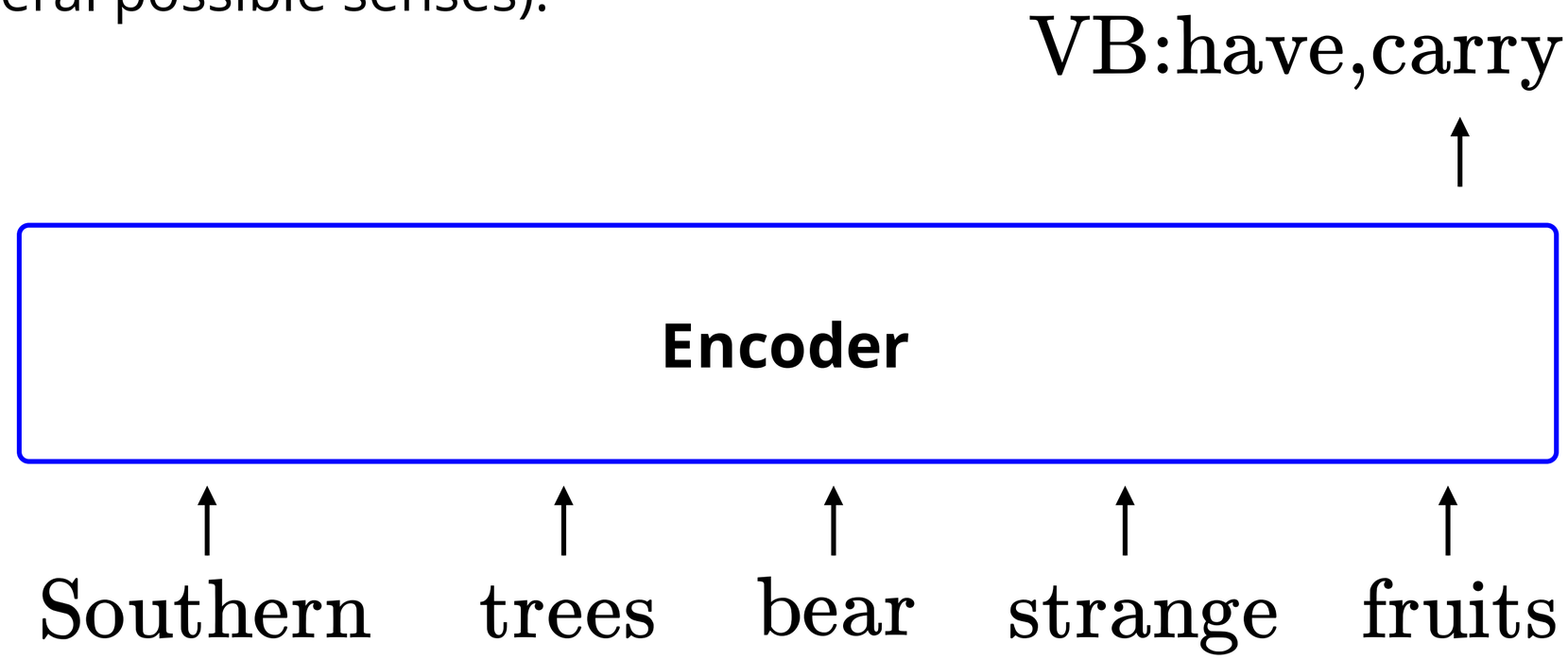
- **Remove the top (prediction) layer corresponding to words** - and replace it with a new layer corresponding to the new task: sequence classification, token classification...
- Train the **full model** (the pre-trained layers + the new prediction layer) **with the new objective**



Fine-tuning an encoder

Fine-tuning:

- **Remove the top (prediction) layer corresponding to words** - and replace it with a new layer corresponding to the new task: sequence classification, token classification...
- Train the **full model** (the pre-trained layers + the new prediction layer) **with the new objective**
- For example, for word sense disambiguation (classification into one of several possible senses):

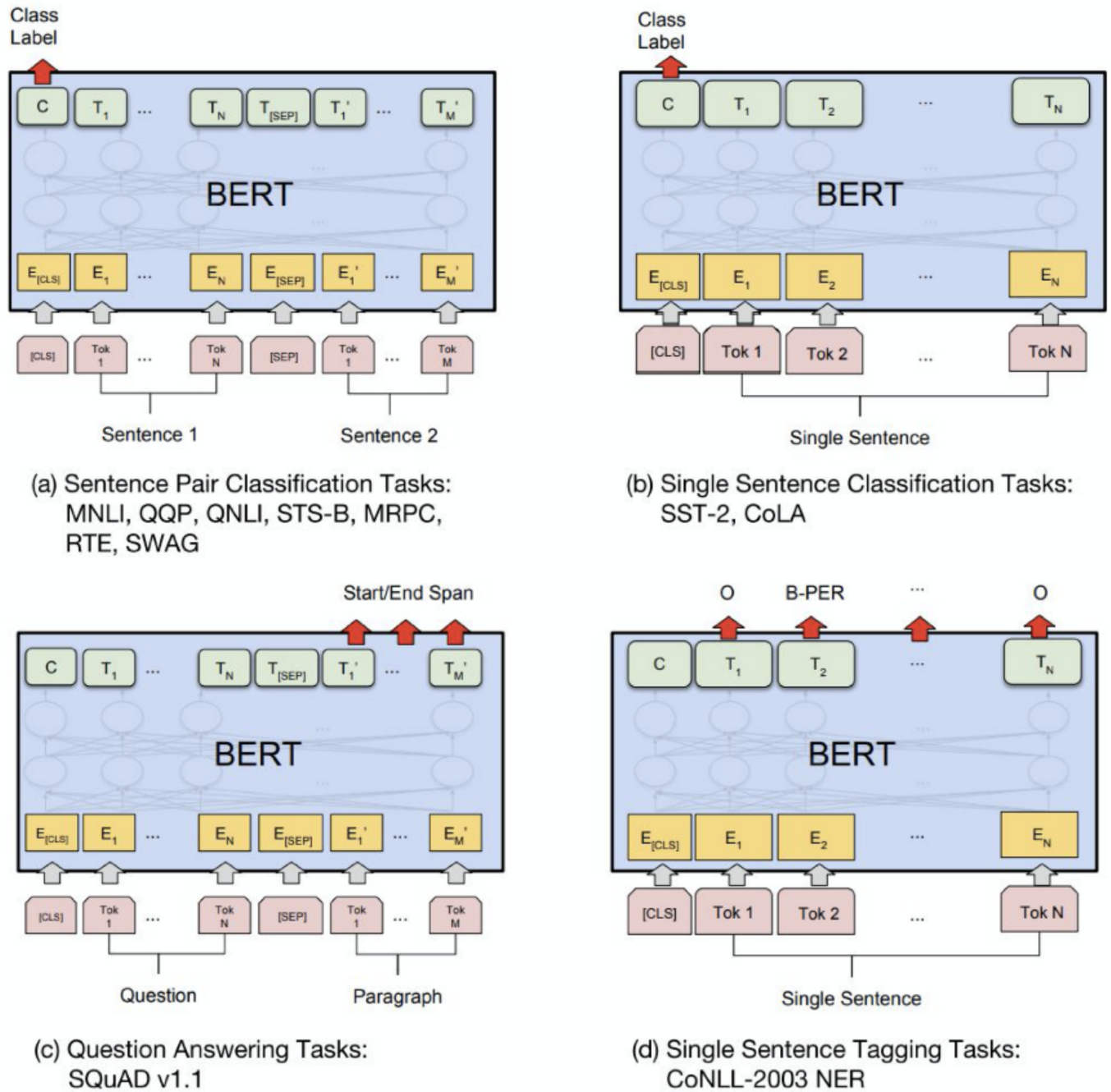


BERT: Fine-tuning and benchmarks

Use appropriate inputs and **change the top layer (head) !**

Beats state-of-the-art on:

- **QQP**: Quora Question Pairs (detect paraphrases)
- **QNLI**: Natural Language Inference
- **SST-2**: Sentiment analysis
- **CoLA**: corpus of linguistic acceptability (detect whether sentences are grammatical.)
- **STS-B**: semantic textual similarity
- **MRPC:** paraphrases dataset
- **RTE:** Natural language inference
- **SQuAD**: Question answering



(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: numerous variants

RoBERTa: *RoBERTa: A Robustly Optimized BERT Pretraining Approach (Liu et al, 2019)*

- Train BERT for longer and and more data
- Findings: simply training more (without changing the architecture) can significantly improve results

BERT: numerous variants

RoBERTa: *RoBERTa: A Robustly Optimized BERT Pretraining Approach (Liu et al, 2019)*

- Train BERT for longer and and more data
- Findings: simply training more (without changing the architecture) can significantly improve results

ALBERT: *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations (Lan et al, 2019)*

- Train BERT using **parameter-reduction techniques**, mainly by factorizing embedding representations and sharing parameters across layers, thus reducing memory use
- Allows to scale up the model, leading to better performance (at the price of a huge computing cost)

BERT: numerous variants

RoBERTa: *RoBERTa: A Robustly Optimized BERT Pretraining Approach (Liu et al, 2019)*

- Train BERT for longer and and more data
- Findings: simply training more (without changing the architecture) can significantly improve results

ALBERT: *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations (Lan et al, 2019)*

- Train BERT using **parameter-reduction techniques**, mainly by factorizing embedding representations and sharing parameters across layers, thus reducing memory use
- Allows to scale up the model, leading to better performance (at the price of a huge computing cost)

SpanBERT: *SpanBERT: Improving Pre-training by Representing and Predicting Spans (Joshi et al, 2019)*

- Masks contiguous spans of tokens instead of individual tokens

- Allows for better results on span-selection tasks

Difficulties with pre-trained encoders

- The model is not **auto-regressive**
  $\rightarrow$ You can't generate sequences efficiently

Difficulties with pre-trained encoders

- The model is not **auto-regressive**
  $\rightarrow$ You can't generate sequences efficiently

- For large models, fine-tuning is **very costly**

Difficulties with pre-trained encoders

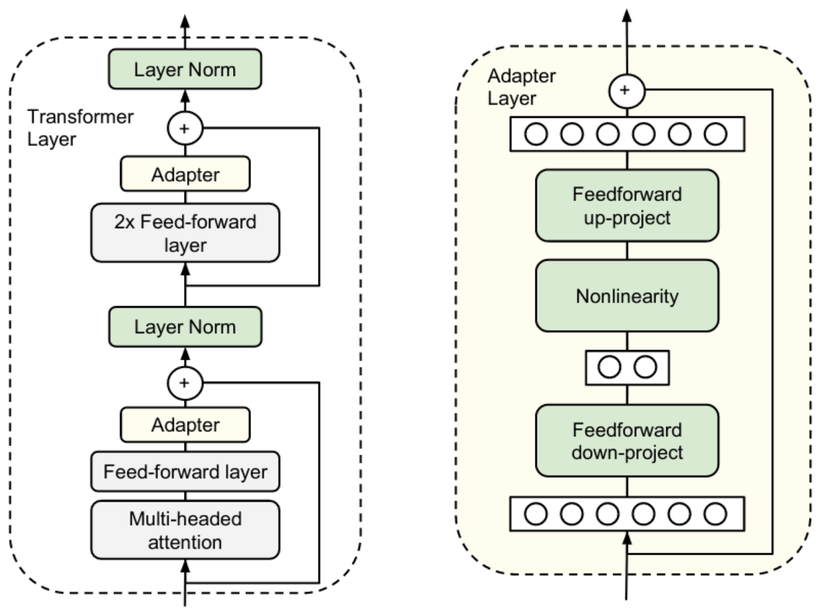
- The model is not **auto-regressive**
  $\rightarrow$ You can't generate sequences efficiently

- For large models, fine-tuning is **very costly**
  - An important direction of research: make it cheap with **parameter-efficient fine-tuning**

Difficulties with pre-trained encoders

- The model is not **auto-regressive**
  $\rightarrow$ You can't generate sequences efficiently

- For large models, fine-tuning is **very costly**
  - An important direction of research: make it cheap with **parameter-efficient fine-tuning**
  - Many approaches:
    - Adapters *(Parameter-Efficient Transfer Learning for NLP, Houlsby et al, 2019)*



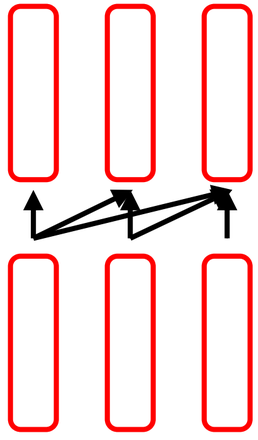
Difficulties with pre-trained encoders

- The model is not **auto-regressive**
  $\rightarrow$ You can't generate sequences efficiently

- For large models, fine-tuning is **very costly**
  - An important direction of research: make it cheap with **parameter-efficient fine-tuning**
  - Many approaches:
    - Adapters *(Parameter-Efficient Transfer Learning for NLP, Houlsby et al, 2019)*
    - Low-Rank Decomposition *(LoRA: Low-Rank Adaptation of Large Language Models, Hu et al, 2021)*

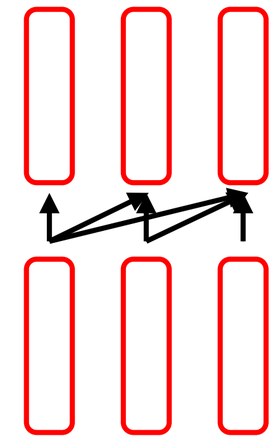
Pre-Training a decoder

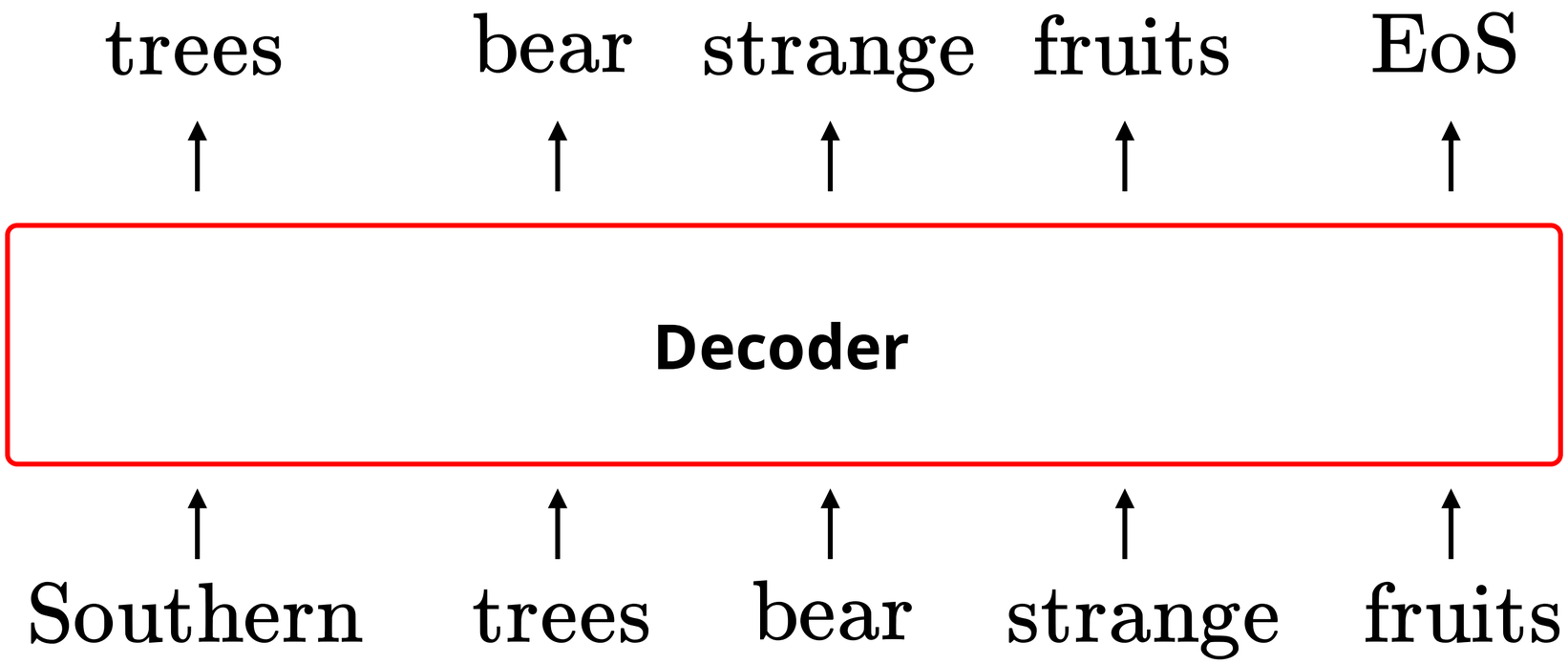
- The decoder is **unidirectionnal:** context from the left
- Here, it's easy, we just use the classical language modeling objective:



Pre-Training a decoder

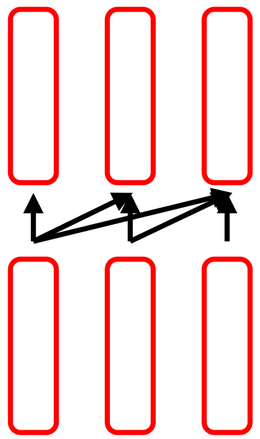
- The decoder is **unidirectionnal:** context from the left
- Here, it's easy, we just use the classical language modeling objective:

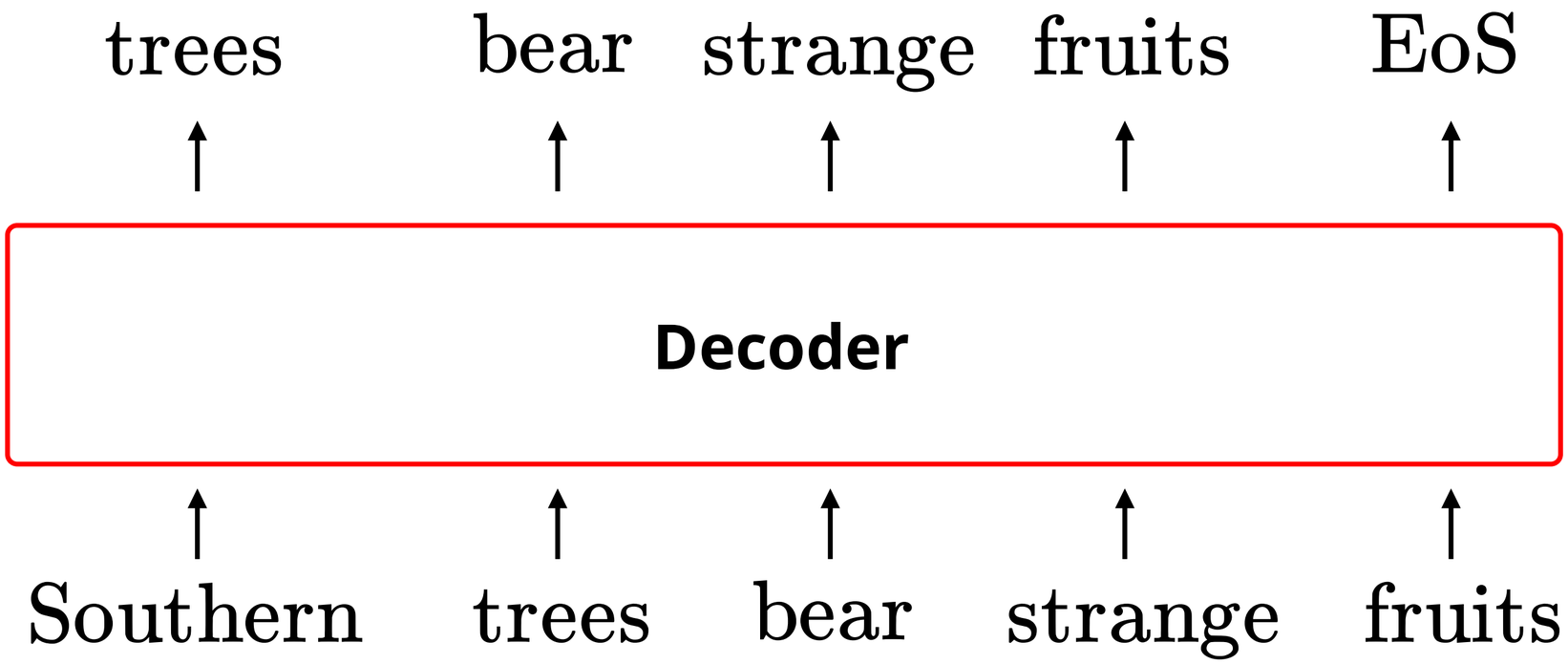




Pre-Training a decoder

- The decoder is **unidirectionnal:** context from the left
- Here, it's easy, we just use the classical language modeling objective:





- We can also **fine-tune a decoder** by removing the prediction layer and replacing it by another one for a new task (*but why, if the goal is to generate text ?*)

Generative Pre-Training

Improving Language Understanding by Generative Pre-Training (Radford et al, 2018)

Generative Pre-Training

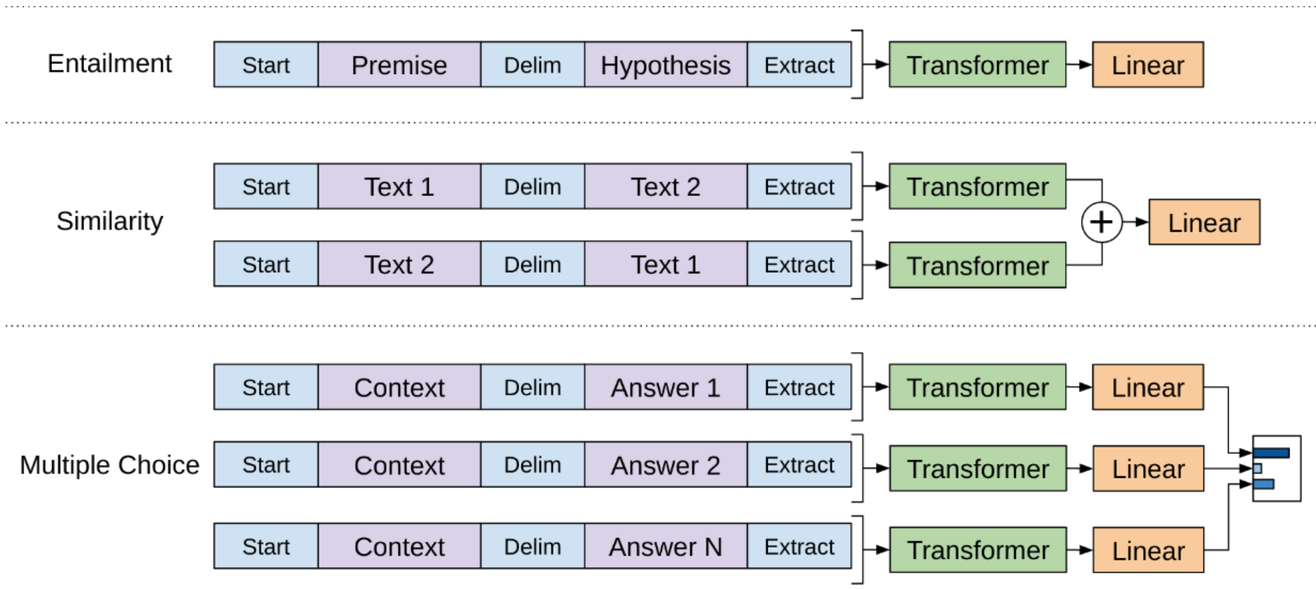
Improving Language Understanding by Generative Pre-Training (Radford et al, 2018)

- **Unsupervised pre-training**: train a 12-stack transformer decoder for a language modeling task, with a vocabulary of size 40.000 (obtained through BPE), on a huge dataset (Bookcorpus, 7000 unique books)

Generative Pre-Training

Improving Language Understanding by Generative Pre-Training (Radford et al, 2018)

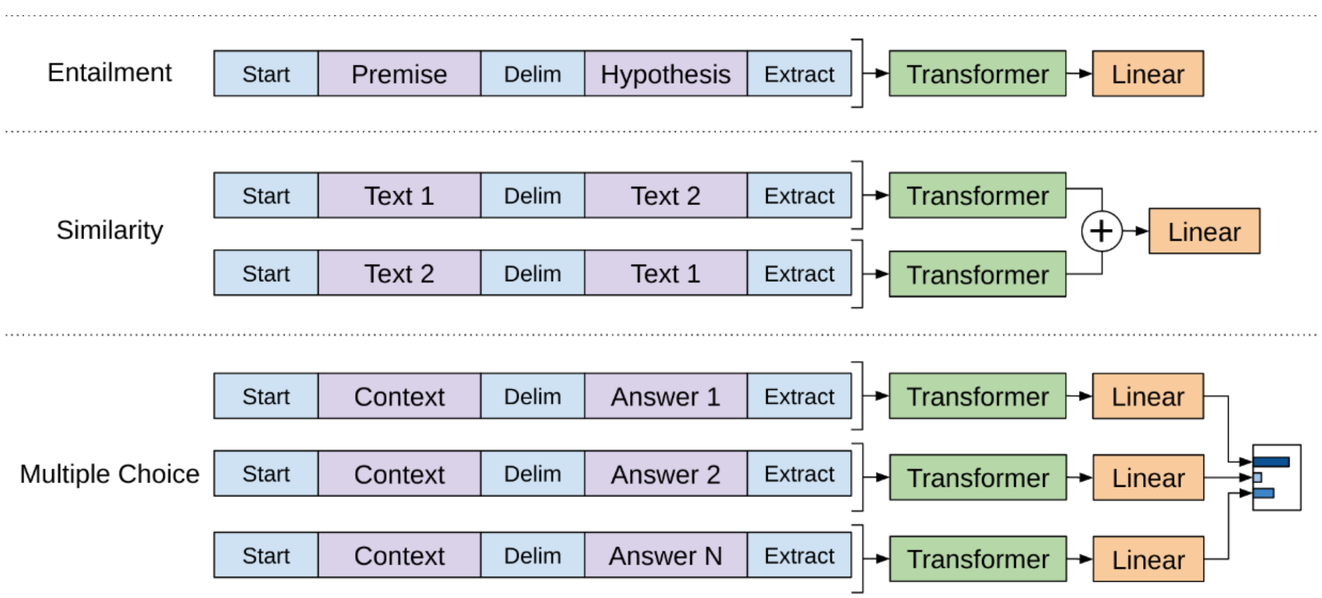
- **Unsupervised pre-training**: train a 12-stack transformer decoder for a language modeling task, with a vocabulary of size 40.000 (obtained through BPE), on a huge dataset (Bookcorpus, 7000 unique books)
- **Supervised fine-tuning**: fine-tune the model on discriminative tasks (*text entailment, similarity, question answering*) with specific formattings:



Generative Pre-Training

Improving Language Understanding by Generative Pre-Training (Radford et al, 2018)

- **Unsupervised pre-training**: train a 12-stack transformer decoder for a language modeling task, with a vocabulary of size 40.000 (obtained through BPE), on a huge dataset (Bookcorpus, 7000 unique books)
- **Supervised fine-tuning**: fine-tune the model on discriminative tasks (*text entailment, similarity, question answering*) with specific formattings:



→ A larger version of GPT, trained on more data, **GPT-2**, produces **relatively convincing samples of natural language** and is very often used as **pre-trained basis** for models which goal will be to generate text.

Abilities of GPT-2

Emergent ability: **Zero-shot learning**

Without supplementary example nor gradient update, performs tasks:

- By using the right format for the input text (*Question Answering*)

Context: Both its sun-speckled shade and the cool grass beneath were a welcome respite after the stifling kitchen, and I was glad to relax against the tree's rough, brittle bark and begin my breakfast of buttery, toasted bread and fresh fruit. Even the water was tasty, it was so clean and cold.
Target sentence: It almost made up for the lack of _____.
Target word: coffee

LAMBADA (Paperno et al, 2016)

- Or by re-framing the task as comparing probabilities of sequences

GPT-2 beats SoTA on benchmark with *no fine-tuning on the task:*

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**

Text-to-Text Transfer Transformer (T5)

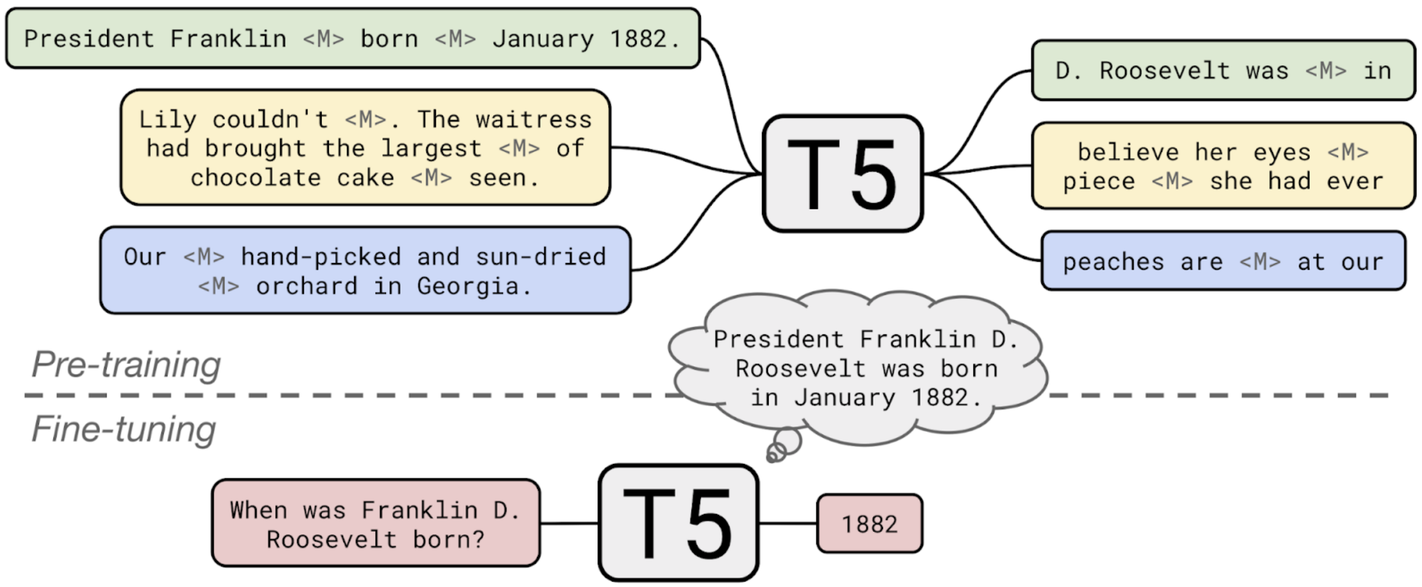
Exploring the Limits of Transfer Learning with a UnifiedText-to-Text Transformer (Raffel et al, 2019)

- Pre-training and fine tuning an **Encoder-Decoder** model

Text-to-Text Transfer Transformer (T5)

Exploring the Limits of Transfer Learning with a UnifiedText-to-Text Transformer (Raffel et al, 2019)

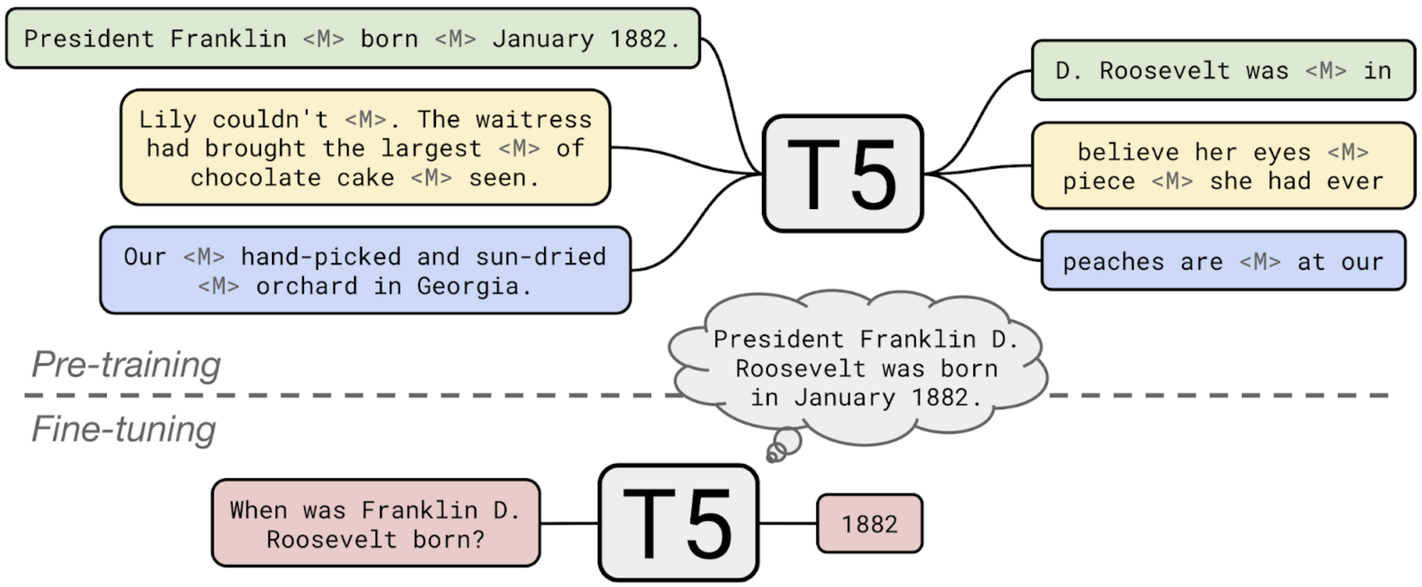
- Pre-training and fine tuning an **Encoder-Decoder** model
- New idea: the model is exclusively **text-to-text**



Text-to-Text Transfer Transformer (T5)

Exploring the Limits of Transfer Learning with a UnifiedText-to-Text Transformer (Raffel et al, 2019)

- Pre-training and fine tuning an **Encoder-Decoder** model
- New idea: the model is exclusively **text-to-text**

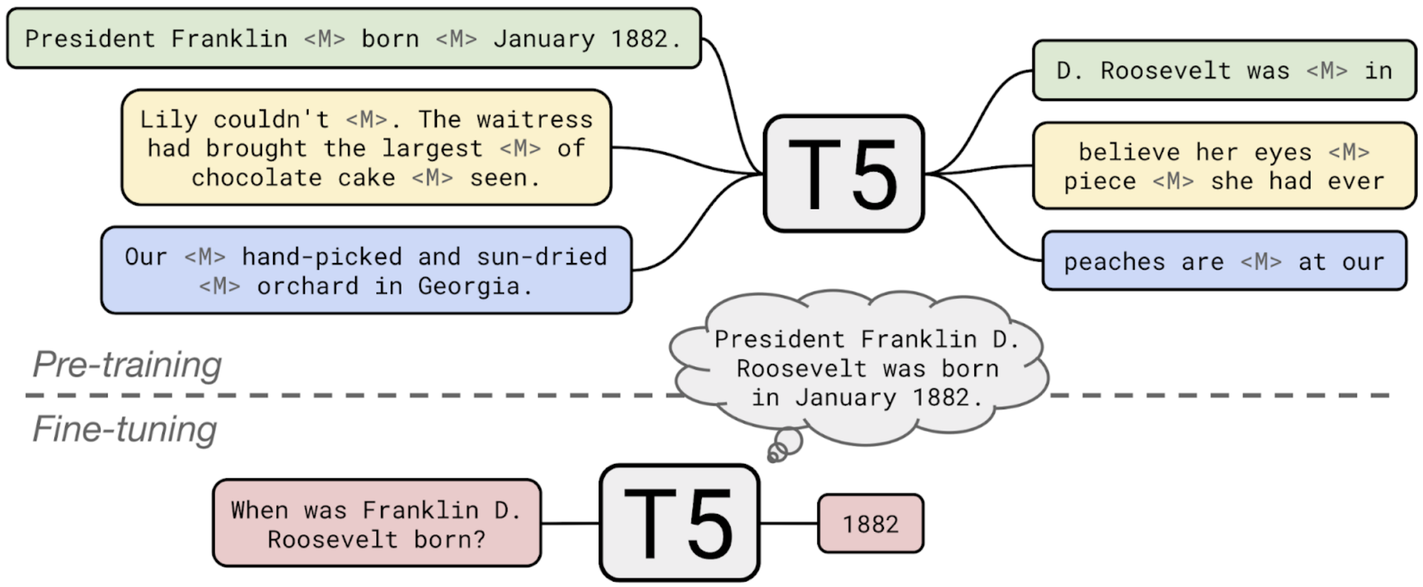


- The model uses and improves on practical methods used in previous models - notably, **span corruption and denoising** for pre-training

Text-to-Text Transfer Transformer (T5)

Exploring the Limits of Transfer Learning with a UnifiedText-to-Text Transformer (Raffel et al, 2019)

- Pre-training and fine tuning an **Encoder-Decoder** model
- New idea: the model is exclusively **text-to-text**



- The model uses and improves on practical methods used in previous models - notably, **span corruption and denoising** for pre-training
- It brought a **new state-of-the-art** on many tasks in the GLUE/SuperGLUE (more difficult language understanding) benchmarks
- Similarly, **standard pre-training basis** for encoder-decoder models

Analyzing NLP models

Model analysis: how ?

How to understand what our models do ?

- **Evaluation** by testing on a few **metrics** on **in-domain test data**
  - Is that enough ? What if the model is good only in this situation ?
  - What if the model uses simple heuristics **unrelated to what we want it to model** ?
- Example of Natural Language Inference (NLI):

  Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference

  (McCoy et al, 2019)

  - Classify a *premise* and *hypothesis* into **entailment**, **contradiction**, or **neutral**
  - Heuristic: **lexical coverage** allows to determine entailment
  - Heuristics are supported by usual datasets...

Challenge sets and Linguistic tests

- Proposal: using **challenging evaluation datasets** testing from specific model skills
- Many evaluation datasets for many aspects of many tasks:
  - Especially popular for Translation and Language Modeling
  - Useful for analysis of biases: *CrowS-Pairs: A Challenge Dataset for Measuring Social Biases in Masked Language Models (Nangia et al, 2020)*
- This last dataset follows the idea of *minimal pairs*: minimal change that would affect the prediction
  - Tests targeting specific errors *Beyond Accuracy: Behavioral Testing of NLP Models with CheckList (Ribeiro et al, 2020)*

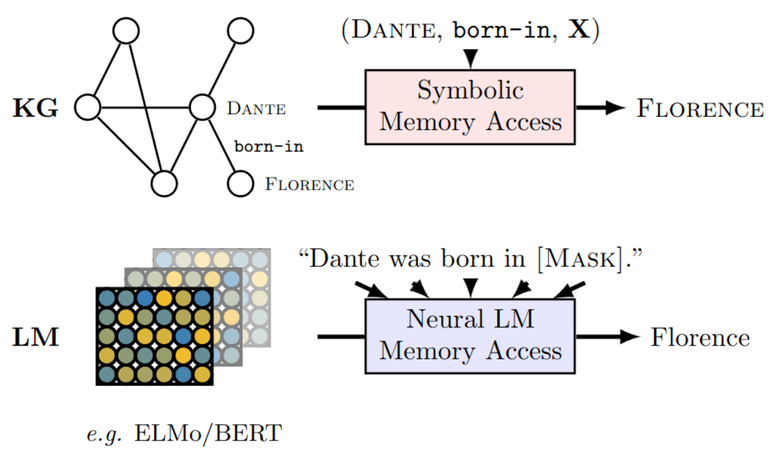
Test case	Expected	Predicted	Pass?
A Testing **Negation** with ***MFT***	Labels: negative, positive, neutral		
`Template: I {NEGATION} {POS_VERB} the {THING}.`			
I can't say I recommend the food.	neg	pos	X
I didn't love the flight.	neg	neutral	X
...			
Failure rate = 76.4%			

B Testing **NER** with ***INV***	Same pred. (`inv`) after removals / additions		
@AmericanAir thank you we got on a different flight to [Chicago → Dallas].	`inv`	pos / neutral	X
@VirginAmerica I can't lose my luggage, moving to [Brazil → Turkey] soon, ugh.	`inv`	neutral / neg	X
...			
Failure rate = 20.8%			

Challenge sets and Linguistic tests

- Many linguistic tests for humans - adapt this to test linguistic capacities of models. For example:
  - What are the grammatical properties that the model is able to capture ? *Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies (Linzen et al, 2016)*
  - What are the linguistic capacities of BERT ? *What BERT Is Not: Lessons from a New Suite of Psycholinguistic Diagnostics for Language Models (Ettinger et al, 2020)*

- Simply retrieve the knowledge that is embedded in models with specifically written prompts: *Language Models as Knowledge Bases (Petroni et al, 2020)*

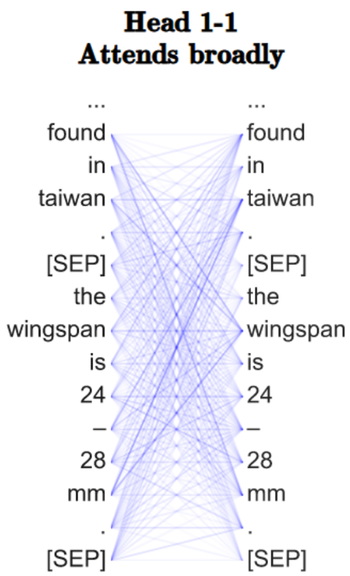


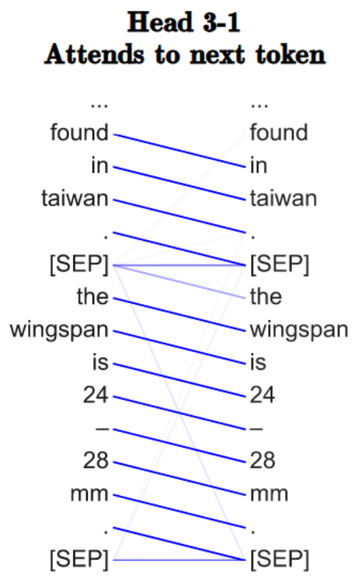
Interpreting model architectures

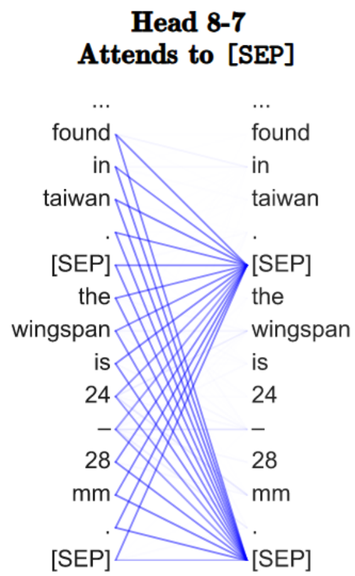
- We have seen that it is possible to interpret **attention** (Translation, sentiment analysis...) - you can also look at some neuron activations
- How does the self-attention behave in BERT ? *What Does BERT Look At ? An Analysis of BERT's Attention (Clark et al, 2019)*

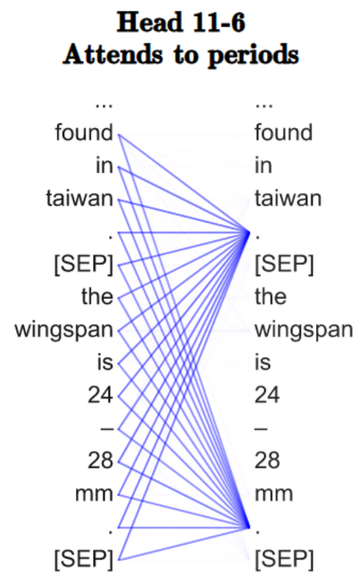
Interpreting model architectures

- We have seen that it is possible to interpret **attention** (Translation, sentiment analysis...) - you can also look at some neuron activations
- How does the self-attention behave in BERT ? *What Does BERT Look At ? An Analysis of BERT's Attention (Clark et al, 2019)*
    - Some attention heads focus on *position*, or *particular tokens* (like [SEP]), or on particular *syntactic functions*



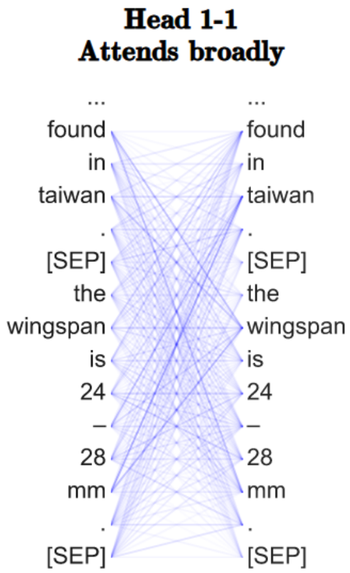


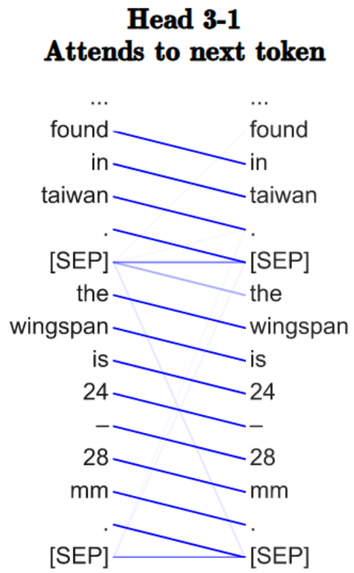


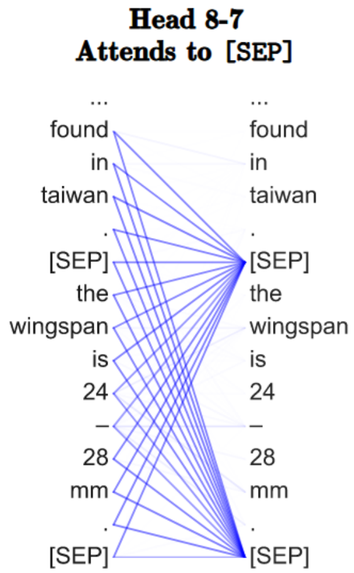


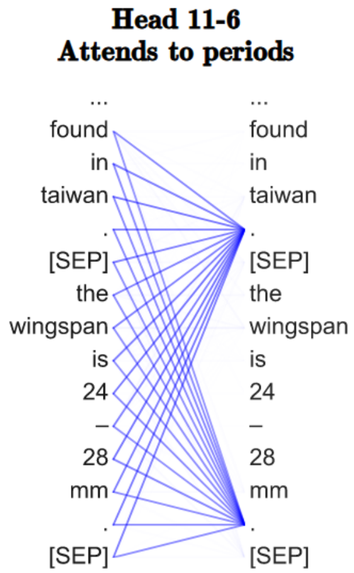
Interpreting model architectures

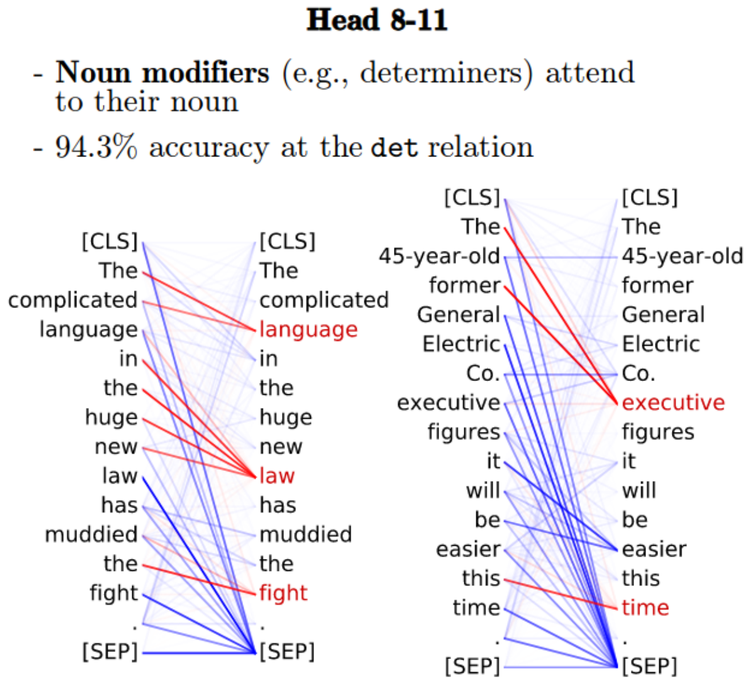
- We have seen that it is possible to interpret **attention** (Translation, sentiment analysis...) - you can also look at some neuron activations
- How does the self-attention behave in BERT ? *What Does BERT Look At ? An Analysis of BERT's Attention (Clark et al, 2019)*
    - Some attention heads focus on *position*, or *particular tokens* (like [SEP]), or on particular *syntactic functions*





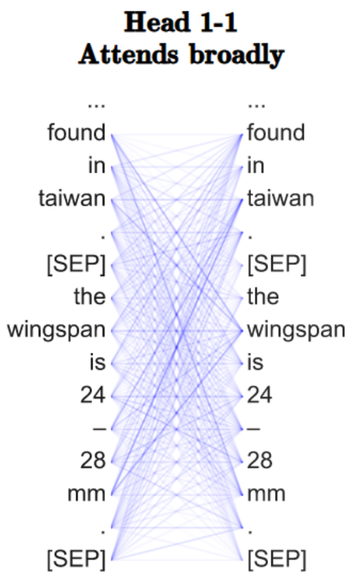


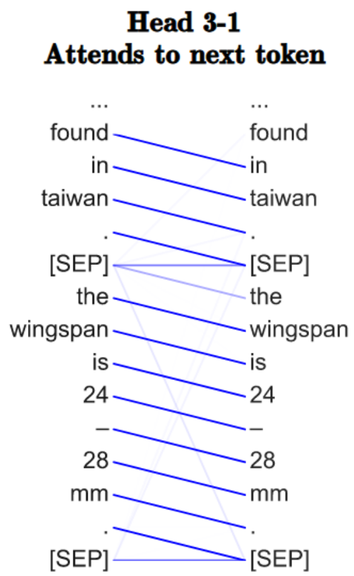


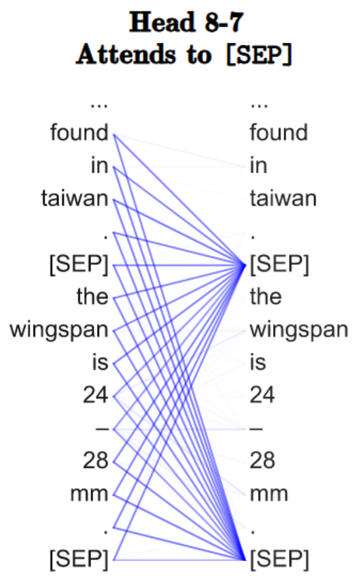


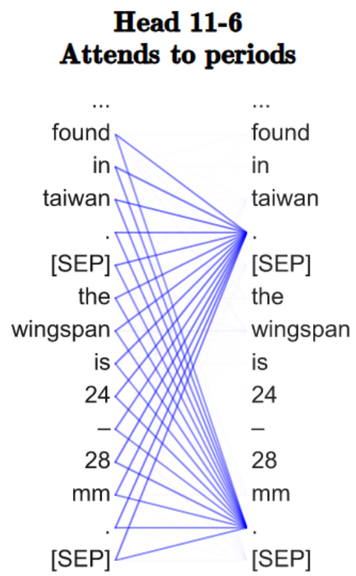
Interpreting model architectures

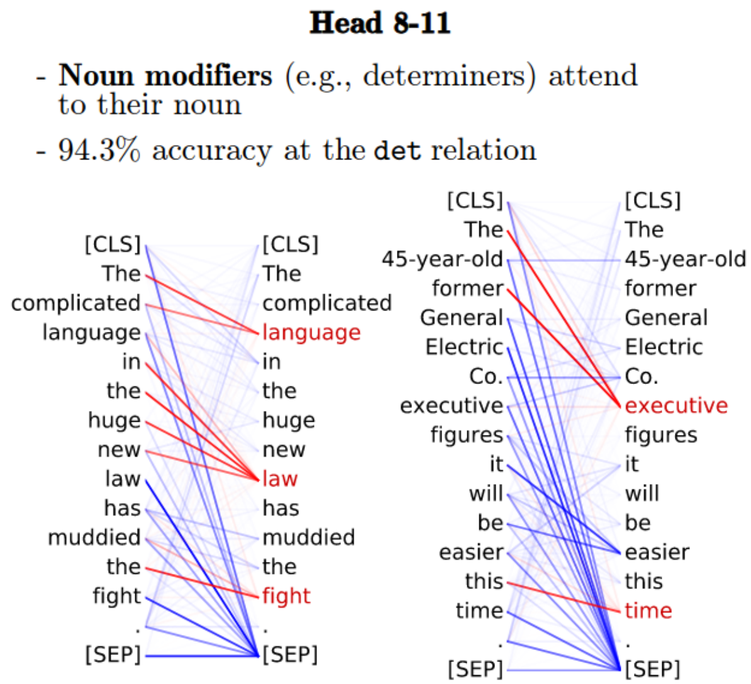
- We have seen that it is possible to interpret **attention** (Translation, sentiment analysis…) - you can also look at some neuron activations
- How does the self-attention behave in BERT ? *What Does BERT Look At ? An Analysis of BERT's Attention (Clark et al, 2019)*
    - Some attention heads focus on *position*, or *particular tokens* (like [SEP]), or on particular *syntactic functions*











→ Hard to interpret and generalize…

Interpreting representations via probing

Probe: a function that we will train to extract a particular linguistic property from a pre-trained representation

Interpreting representations via probing

Probe: a function that we will train to extract a particular linguistic property from a pre-trained representation

- Usually, a **simple linear classifier**

Interpreting representations via probing

Probe: a function that we will train to extract a particular linguistic property from a pre-trained representation

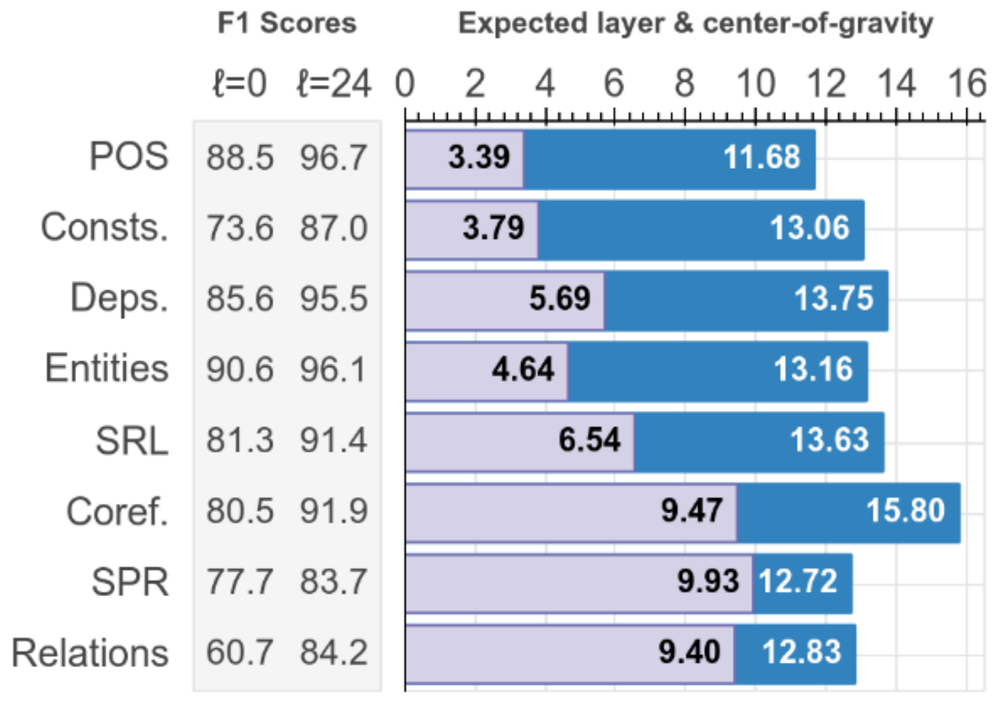
- Usually, a **simple linear classifier**
- Measure how *accessible* is that property in the representation

Interpreting representations via probing

Probe: a function that we will train to extract a particular linguistic property from a pre-trained representation

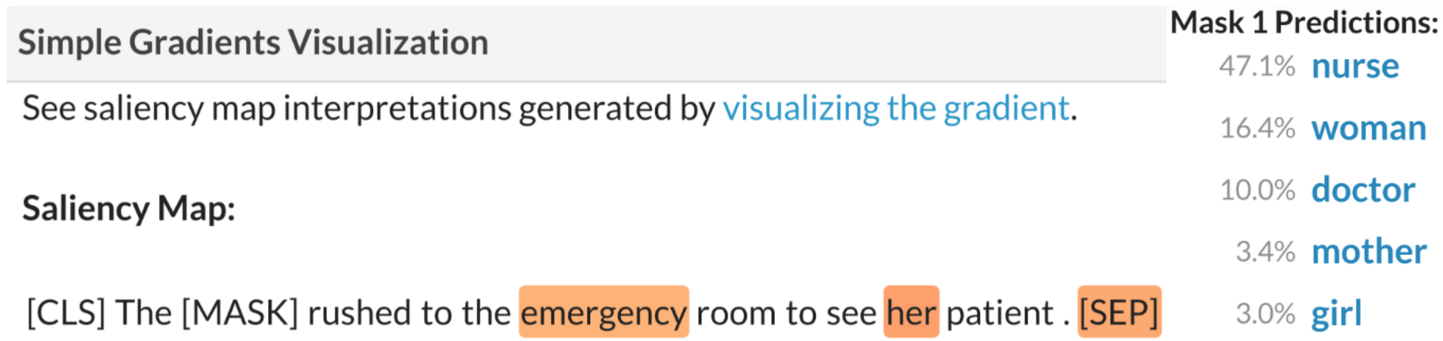
- Usually, a **simple linear classifier**
- Measure how *accessible* is that property in the representation
- Example: from which layer are the usual properties of a classical NLP pipelines better predicted ?

BERT Rediscovers the Classical NLP Pipeline (Tenney et al, 2019)



Explaining model predictions

- Study what, in the input, led to the prediction:
  - Word prediction task where *long* context is necessary *The LAMBADA dataset: Word prediction requiring a broad discourse context (Paperno et al, 2016)*
  - Inversely, try to remove distant context - *what changes ?* *Sharp Nearby, Fuzzy Far Away: How Neural Language Models Use Context (Khandelwal et al, 2018)*
  - Use **saliency methods** *AllenNLP Interpret: A Framework for Explaining Predictions of NLP Models (Wallace et al, 2019)*



- Again, use minimal pairs: **adversarial examples**
  - Semantically equivalent queries to probe for incorrect answers *Semantically Equivalent Adversarial Rules for Debugging NLP Models (Ribeiro et al, 2018)*
  - Devise contrast sets to challenge models *Evaluating Models' Local Decision Boundaries via Contrast Sets (Gardner et al, 2020)*

Large Language Models: efficient adaptation

Large Language Models: a summary

	Encoders	Decoders	Encoder-decoders
Formally:	**NOT** LMs	LMs	
Referred as:	*Masked* LMs	*Causal* or *auto-regressive* LMs	
Examples:	BERT, XLNet, ERNIE,...	GPT-1, 2, ...	BART, T5, LaMDA

Generative models become the focus,

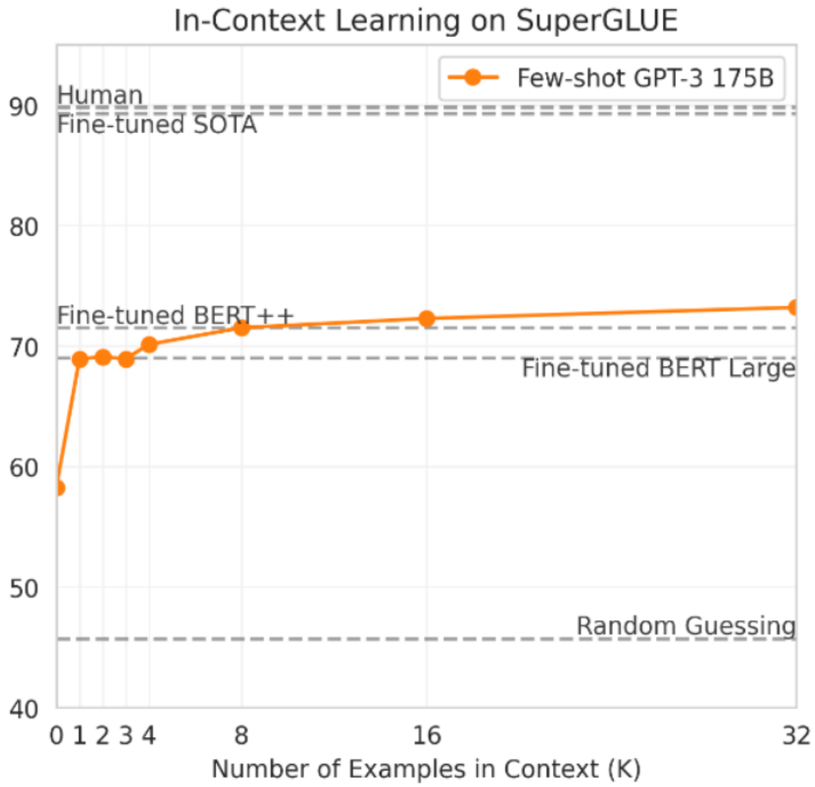
mainly decoder-only models are scaled-up

So now, we talk about: GPT-3, PaLM, BLOOM, LLaMA, Falcon, Mistral, ...

GPT-3, a very large language model

Language Models are Few-Shot Learners (Brown et al, 2020)

- The largest T5 model has 11 billion parameters; however, the GPT-3 model has **175 billions**

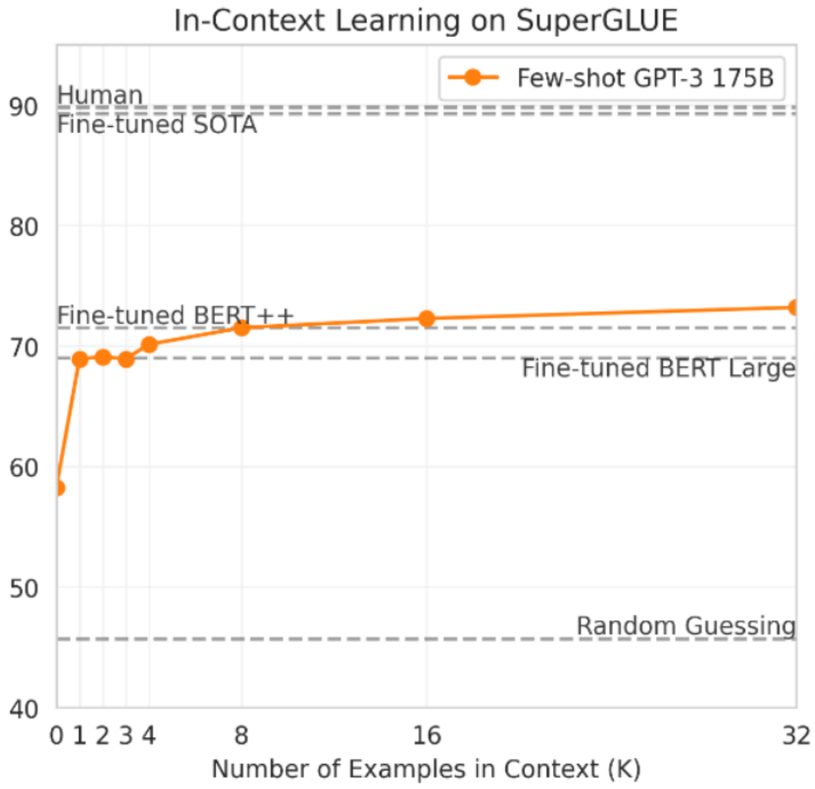


← Few-shot learning is on par with large specialized models

GPT-3, a very large language model

Language Models are Few-Shot Learners (Brown et al, 2020)

- The largest T5 model has 11 billion parameters; however, the GPT-3 model has **175 billions**
- GPT-3 shows a behavior that has been labelled as "**in-context learning**", where the model is able to perform zero-shot learning by being given a natural language description of the task:



← Few-shot learning is on par with large specialized models

Reverse words in a sentence

This is great
Great is this

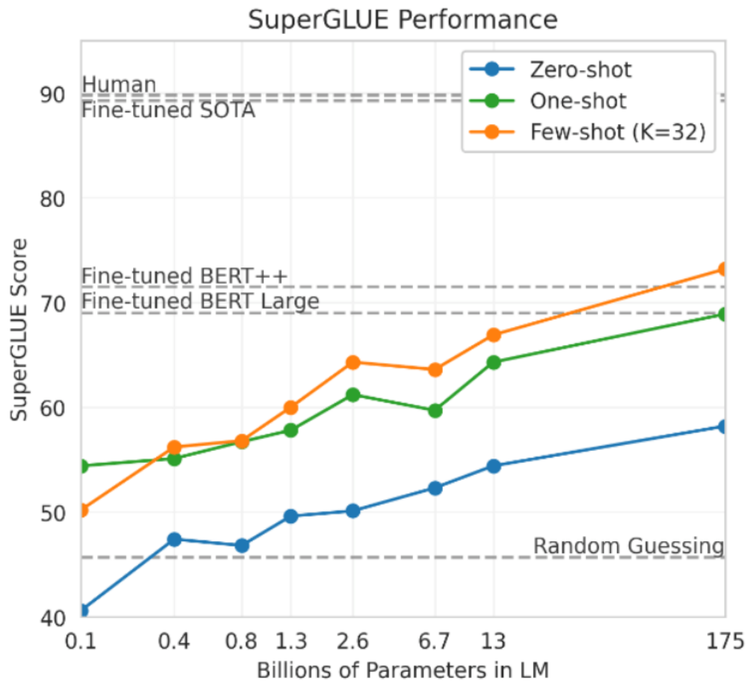
The man on the moon
Moon the on man the

Will this really work
Work really this will

I hope this is a big achievement
Achievement big I hope this is

The king came home on a horse
Home horse king came the

New usage: prompting



← With model scale, **in-context few-shot learning** becomes an emergent property

More attractive than *supervised* fine-tuning: gradient updates necessitate **more examples** and **computing power**

Limitations of *prompting* at the time:

- Limited by context size (while large, issue for some tasks)
- Tasks involving richer, multi-step reasoning

  → A solution: *chain-of-thought prompting*

  Chain-of-Thought Prompting Elicits Reasoning in Large Language Models (Wei et al, 2022)

  Works better for larger models !
- Overall, prompt-based learn is sensitive and inefficient

  Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? (Min et al, 2022)

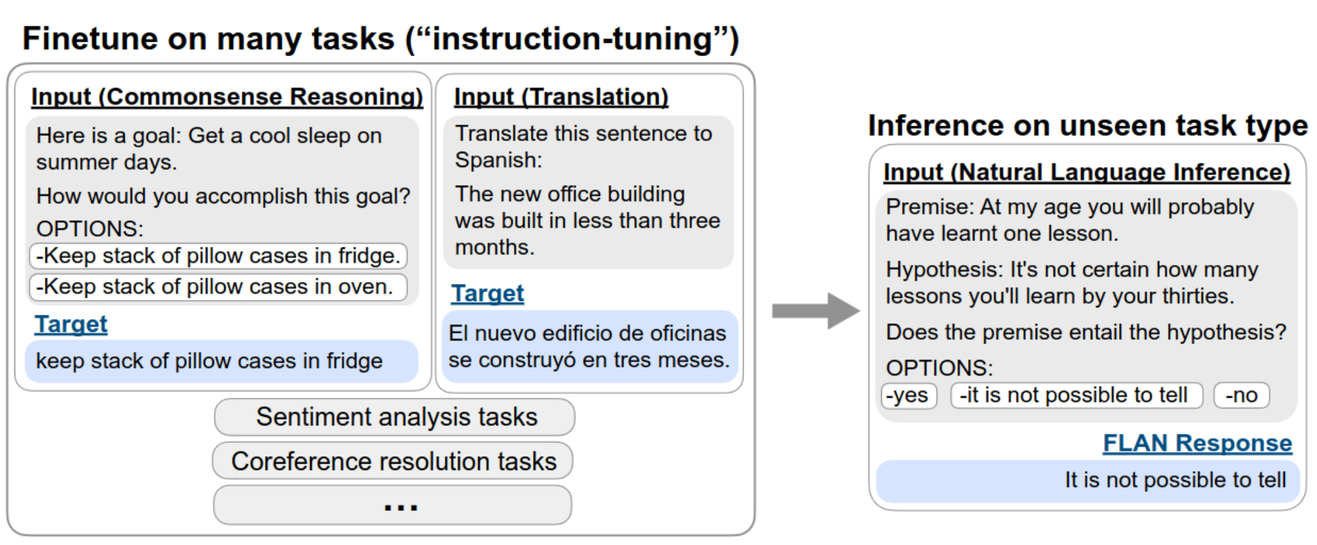
Parameter Efficient Fine-Tuning

- **Fine-tuning**: training all parameters of the model
- Parameter Efficient Fine-tuning (**PEFT**): training a small subset of the parameters
- Necessary because of **model size** + **diversity of tasks**
- See "**Modular and Parameter-Efficient Fine-Tuning for NLP Models**" (EMNLP 2022 Tutorial)
  - Parameter composition (e.g, Low-rank Adaptation (LORA))
  - Input composition (e.g, prefix tuning)
  - Function composition (e.g, Adapters)

Instruction fine-tuning

Goal: improve ability to do zero-shot learning
(Especially when training data does not contain examples of the task)

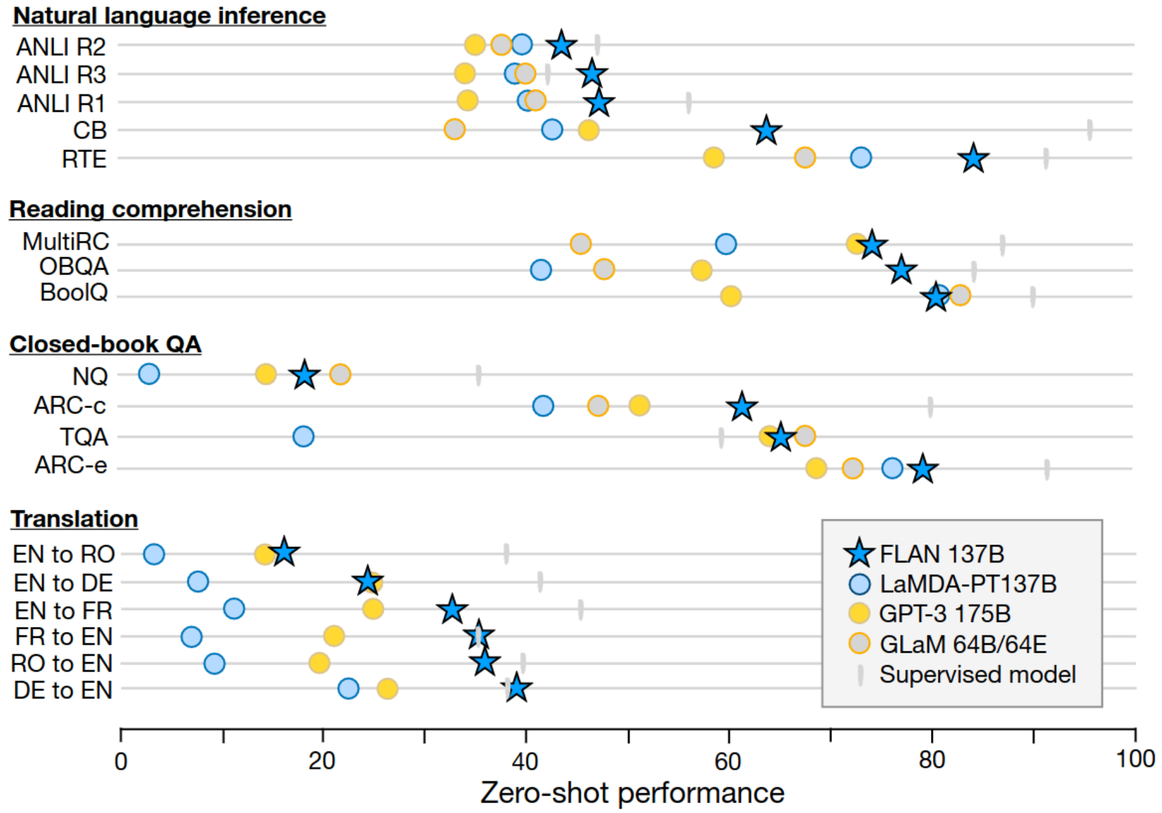
- The idea is to train the model to generalize to **unseen tasks**
- Intuition: describe the task via **natural language instructions**



Fine-Tuned Language Models are Zero-shot Learners (Wei et al, 2022)

- Collect examples of (*instructions, output*) pairs across many tasks and fine-tune the model
  - Evaluate on **unseen tasks**

Instruction fine-tuning: evaluation



- Very encouraging results !
- Again, the key seems to be scaling up. Later:
  - *Scaling Instruction-Finetuned Language Models (Chung et al, 2022)* → 1,8k tasks !
  - Large multitasks evaluation benchmarks like **MMLU** (*Hendrycks et al, 2021*) and **BBH** (*Srivastava et al, 2022*)

Limitations of instruction fine-tuning

- Collecting ground truth is **expensive**
- More difficult: some tasks (implying *open-ended* generation) may have **no right answer**
  → *'Write me a story about ....'*
- Another issue: **no ordering of mistakes**, while some are of course worse than others
- Lastly: still, **mismatch** between the training objective and 'following the user's instructions'

Next: Is it possible to **explicitly** satisfy human preference ?

- How to design objectives for user intent (*implicit* understanding) ?
- *Scalable agent alignment via reward modelling (Leike et al, 2018)*
  - Idea: learning a **reward function** from **interaction** with the user
  - Optimize with the learned reward function with reinforcement learning

Aligning LLMs to their human users

Integrating Human Feedback in LMs

Training language models to follow instructions with human feedback (*Ouyang et al, 2022*)

- Goal: **Aligning language models to their users' intent**

What is *Alignment* ?

- Language modelling objective $\neq$ "follow the user's instructions"
  $\rightarrow$ It is **misaligned**
- User's intentions ?
  - *Explicit*: following instructions
  - *Implicit*: truthful, not toxic, not biased, nor harmful
    $\rightarrow$ **HHH**: helpful, honest, harmless (*Askel et al, 2021*)
- Alignment using **reward modeling** (*Leike et al, 2018*) and reinforcement learning

Optimizing for human preferences

Assuming a *human reward* R of a task, e.g summarization

SAN FRANCISCO, California (CNN) -- A magnitude 4.2 earthquake shook the San Francisco ... overturn unstable objects.	An earthquake hit San Francisco. There was minor property damage, but no injuries.	The Bay Area has good weather but is prone to earthquakes and wildfires.
	s_1	s_2
	$R(s_1) = 8.0$	$R(s_2) = 1.2$

From Stanford's class CS224N - Lecture 11

- The goal is to maximize the expected reward of samples from the Language model:

$$\mathbb{E}_{\hat{s} \sim p_\theta}[R(\hat{s})]$$

- Appropriate tool: **Reinforcement Learning**
  - Advances in RL that work for large neural models are used (*Proximal Policy Optimization Algorithms, Schulman et al, 2017*)

Modelling human preferences

- First difficulty: **interaction** with human will be **costly**
  - Solution: create a **reward model** of their preferences
  - From an annotated dataset of $(x, s, R(s))$, fine-tune a LM into a reward model $r_\phi(x, s)$ to predict $R(s)$
  - Then, optimize for $\mathbb{E}_{\hat{s} \sim \mathbb{P}_\theta}[r_\phi(x, \hat{s})]$ instead

- Second difficulty: how to design a suitable R ?
  - *Deep Reinforcement Learning from Human Preferences (Christiano et al, 2017)*: ask humans for **pairwise comparisons**
  - Use the ***Bradley-Terry*** *(1952)* paired comparison model, where s^w is the *winning* sample and s^l the *losing* sample:

$$\mathbb{E}_{(s^w, s^l) \sim D}[\log \sigma(r_\phi(x, s^w) - r_\phi(x, s^l))]$$

Reinforcement Learning Human-Feedback

- Now, assuming that we have:
  - A pre-trained language model p^{PT} (which could also be instruction fine-tuned, p^{SFT})
  - A dataset D of human comparisons, on which we trained a reward model r_ϕ initialized from p^{PT}

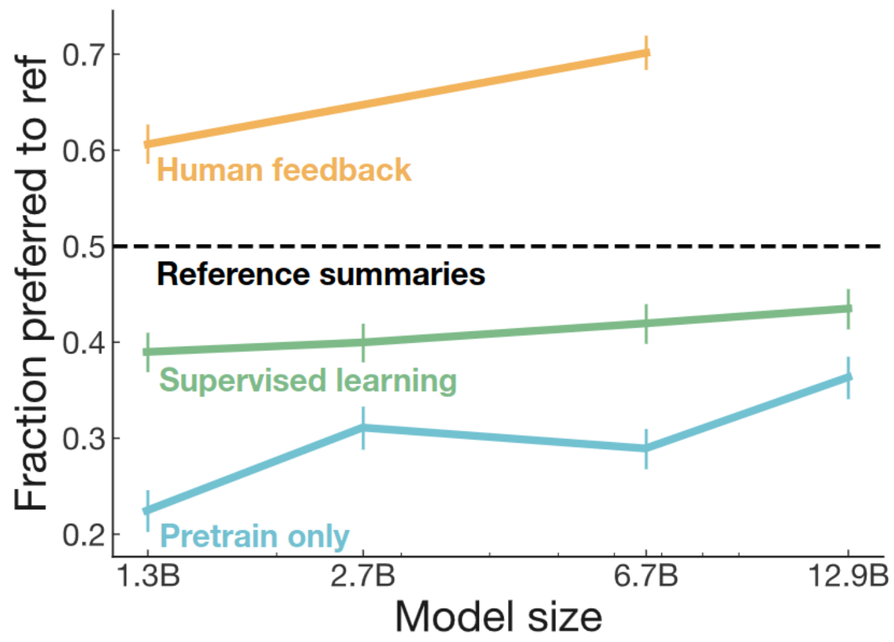
- **Applying RLHF**: *Fine-tuning Language Models from Human Preferences (Ziegler et al, 2029*)
  - Initialize a model p_θ^{RL} from p^{PT}
  - Optimize a reward R *penalized for diverging from the* PT *model* with **Proximal Policy Optimization** (PPO)

$$R(s) = r_\phi(x, s) - \beta \log \frac{p_\theta^{RL}(s)}{p^{PT}(s)}$$

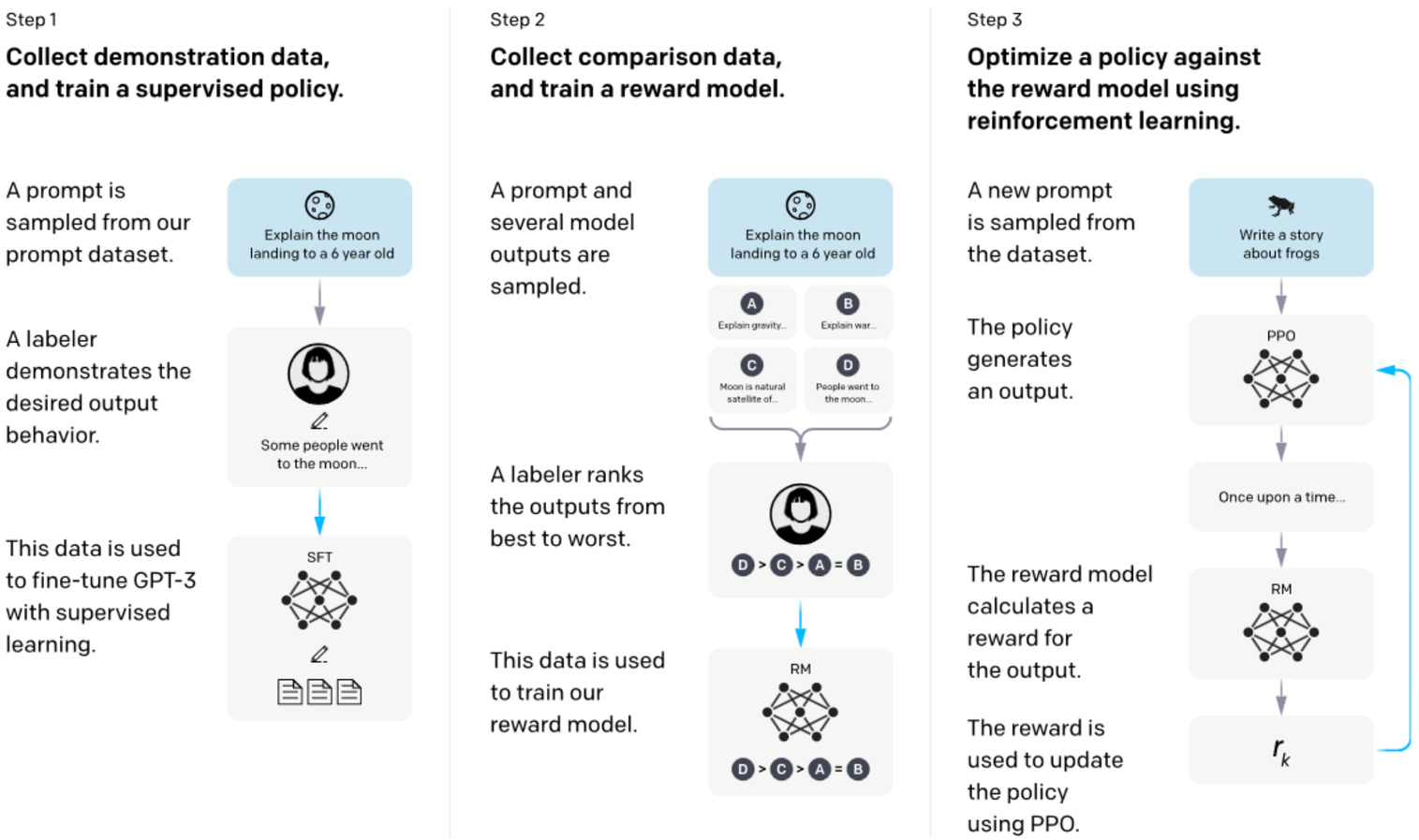
In expectation, corresponds to KL divergence

RLHF: First results

- *Ziegler et al, 2019*: Application to 4 tasks
  - **Mixed results** depending on the task
  - Challenges: labelling for this is hard, data collection is hard
  - The KL penalty is necessary
- *Learning to summarize from human feedback* (*Stiennon et al, 2020*)
  - Larger models, better data collection + minor modifications
  - **Significant gain** in performance
  - **Better generalization** to new domains



Scaling up the tasks: InstructGPT



- **Main difference:** data collection and labelling !
  - Use data from an hosted API and ~40 contractors for producing and labelling prompts
  - The model is trained on a huge variety of tasks (13K for SFT, 33K for RM, 31K for PPO)

Scaling up the tasks: InstructGPT

- **Human data collection:**
  - Several criteria for labeler selection, based on agreement with *labels from the research team* on rankings and sensitive content
  - Detailed instructions for labelling - again, following *Askell et al (2021)*: ensure that outputs are **helpful**, **truthful** and **harmless**
- **Tasks:**
  - Initial prompts written by the labeler team
  - Early version of the model on the API used to gather more prompts
  - Various types of prompts:

    - **Plain:** We simply ask the labelers to come up with an arbitrary task, while ensuring the tasks had sufficient diversity.
    - **Few-shot:** We ask the labelers to come up with an instruction, and multiple query/response pairs for that instruction.
    - **User-based:** We had a number of use-cases stated in waitlist applications to the OpenAI API. We asked labelers to come up with prompts corresponding to these use cases.

InstructGPT: models

- **Models:**
  - Pretty much what we saw right before *(Stiennon et al, 2020)*
  - Initial results show PPO reduces performance on 'classical' NLP benchmarks - the paper proposes to mix-in gradient from the pre-training loss to fix it: **PPO-ptx** adds the following term

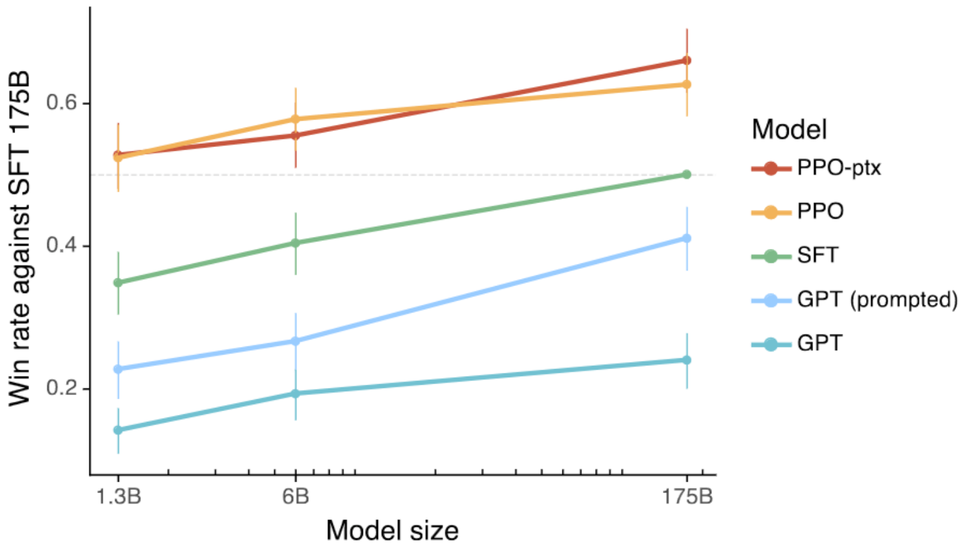
$$\gamma \mathbb{E}_{s \sim D_{pre-train}} [\log(p_\theta^{RL}(s))]$$

  - Baselines are:
    - **GPT-3**
    - GPT-3 with supervised instruction fine-tuning (**SFT**) - the result of Step 1
    - **GPT-3 prompted** with a few-shot 'prefix'
    - **GPT-3 fine-tuned** on public instructions dataset (T0 and FLAN)

InstructGPT: evaluation

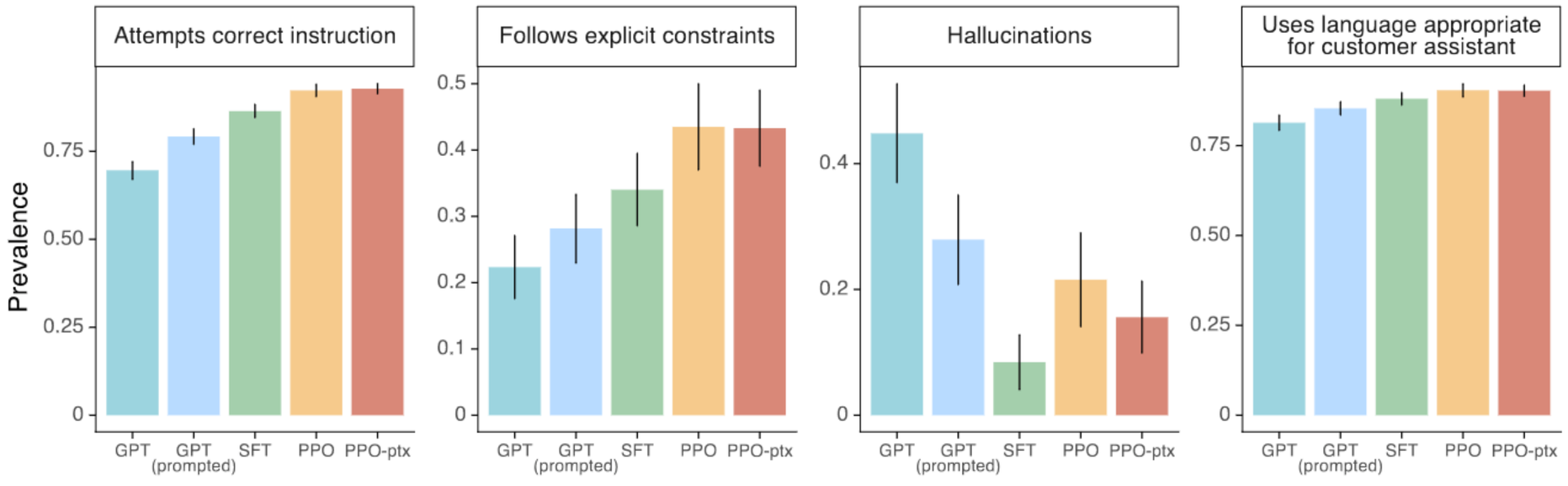
- **Evaluation:**
  - **Truthfulness**: use closed-domain tasks to detect *hallucinations*, and evaluate on the TruthfulQA dataset *(Lin et al, 2021)*
  - **Harms**: benchmarks for bias and toxicity, like RealToxicityPrompts (*Gehman et al, 2020*) and CrowS-Pairs (*Nangia et al, 2020*)
    → *The three datasets contain prompts which the result of for each model will be labelled by humans*
  - **Helpfulness**: labeler preference ratings
    - Use both prompts submitted to the early InstructGPT API and the GPT-3 API to avoid unfairness
  - Plus, some analysis of qualitative results

InstructGPT: results

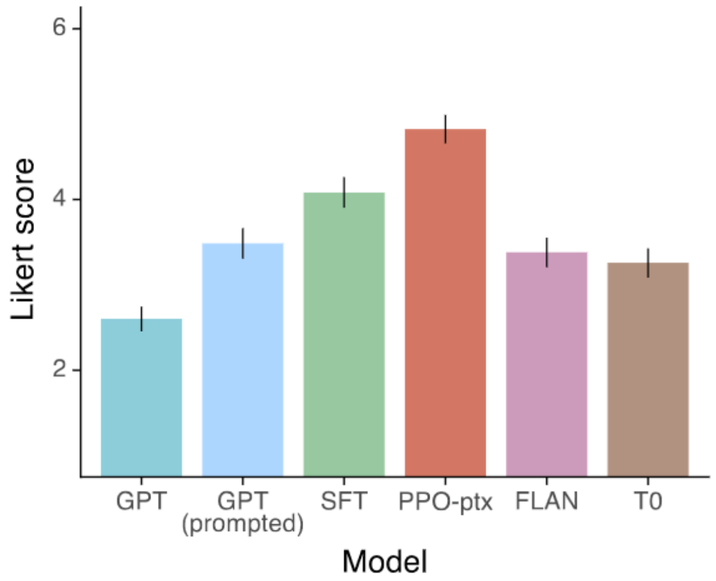


← Comparison of approaches on **human preference** (*vs the 175B SFT model*) on API prompt distribution

Various human ratings of approaches (across model sizes) ↓



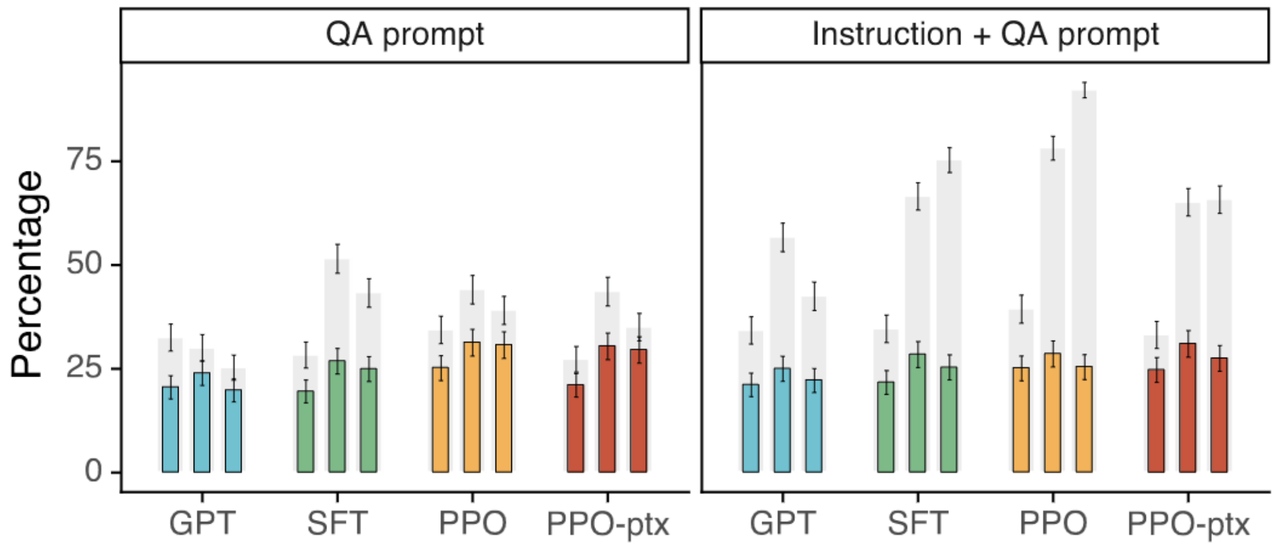
InstructGPT: results



← Likert ratings by humans on some of the models, added those fine-tuned on public datasets

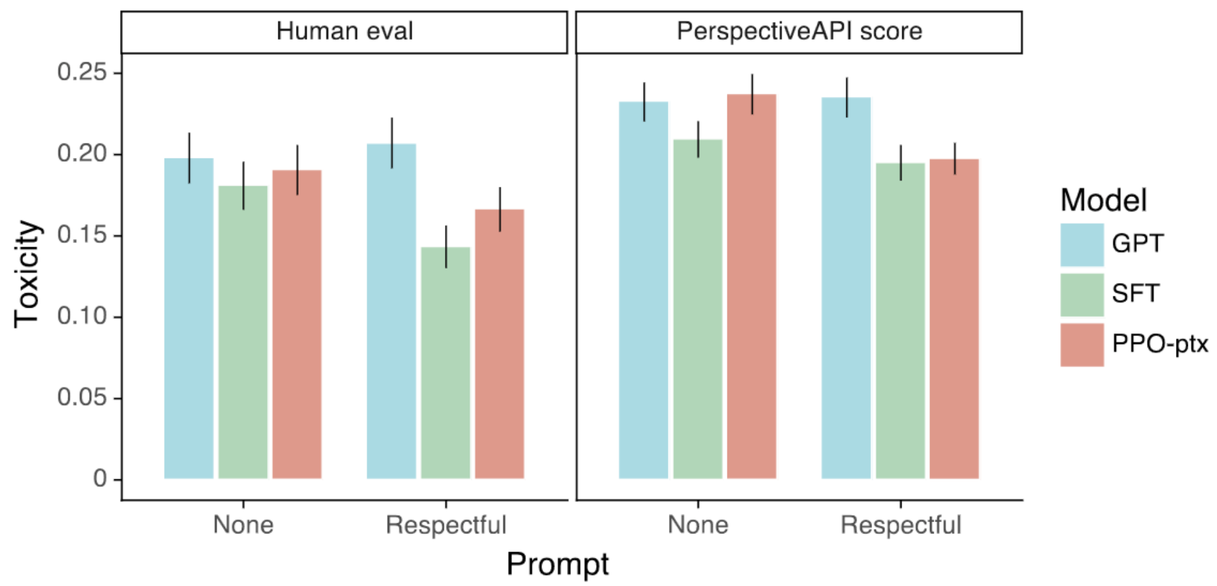
- Overall, key results on **human preferences:**
  - **Significant preference for Instruct GPT** over GPT-3
    - This is maintained over GPT-3 prompts, and on 'held-out' labelers that did not produce training data
  - Models fine-tuned on **public datasets are outperformed**
    - Hypothesis: their design is not align with what users ask + their data is less diverse

InstructGPT: results



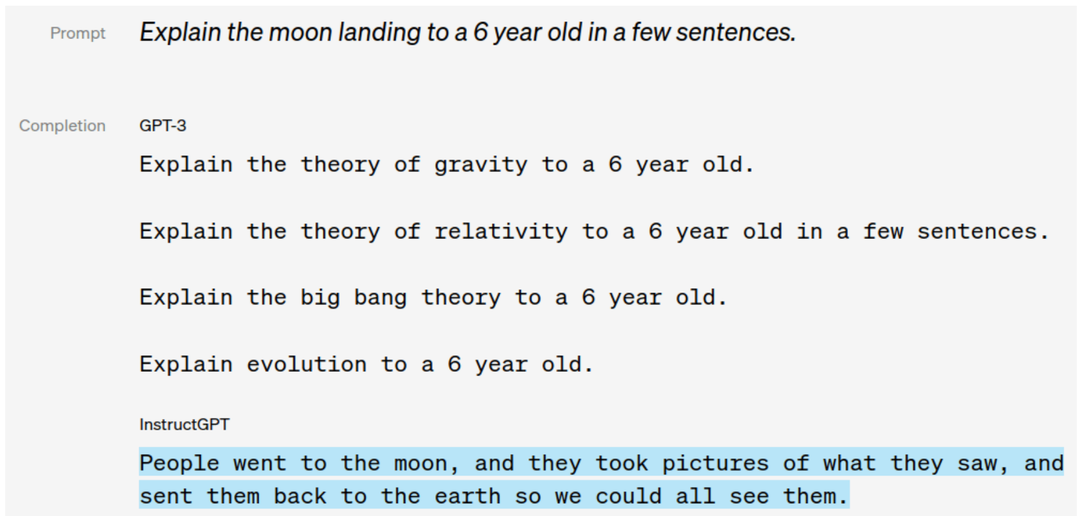
← On TruthfulQA, ratings of truthfulness (*gray*) and truthfulness and informativeness (*color*) : **small improvement**

On RealToxicityPrompts, ratings by humans and by a dedicated API - **small improvement obtained by using instructions** →



InstructGPT: results

- **Supplementary results**:
  - The modified objective PPO-ptx does help minizing performance regression on public NLP datasets
  - InstructGPT seems to **generalize very well to new tasks**



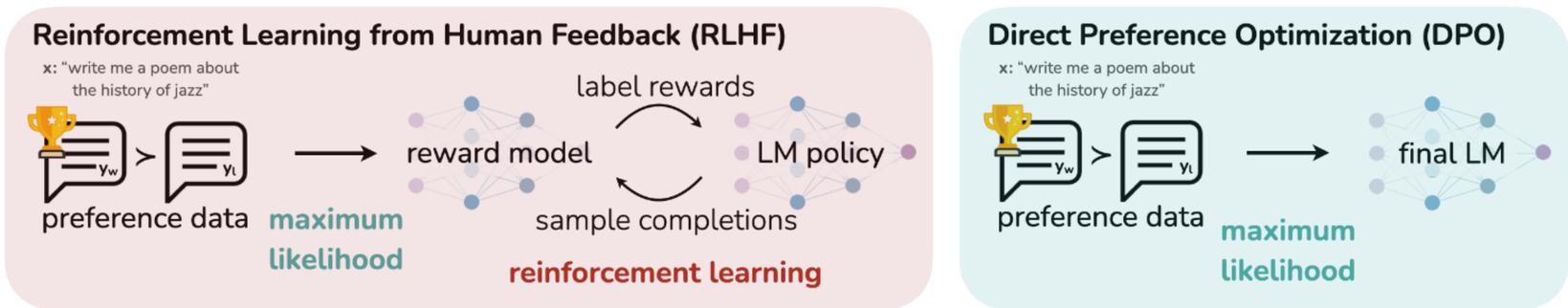
  - InstructGPT still makes simple mistakes
    - Sometimes does not detect false premise
    - Does not manage well several constraints in the instruction
    - Sometimes is indecisive, avoid clear answers

InstructGPT: discussion and limitations

- Important point: the **computational cost** of RLHF is modest compared to pre-training
- Remaining question: do **improvements to unseen tasks scale** with model size, quantity of training data ?
- Limitations for **alignment**:
  - Small team of labelers, no redundancy on labelling
  - Aligning to labelers, and by proxy to the team of researchers
  - Training data limited to labelers + early OpenAI customers
  - Progress to make on the notion of alignment itself: what should it be (rather than what labelers guessed of user preference) ?
- Overall, the biggest issues are still there:
  - RL is difficult
  - Human preferences are unreliable; models of them even more; $\rightarrow$ clear possibility of **hallucination** and **misalignment**

InstructGPT: aftermath

- **ChatGPT** : trained with supervised instruction fine-tuning and RLHF on **dialog tasks**
- A lot of work on improving RLHF:
  - Some of it towards lowering data requirements, notably with using the model's own outputs (*Huang et al, 2022*)
  - Notably: **Direct Preference Optimization** (*Rafailov et al, 2023)*



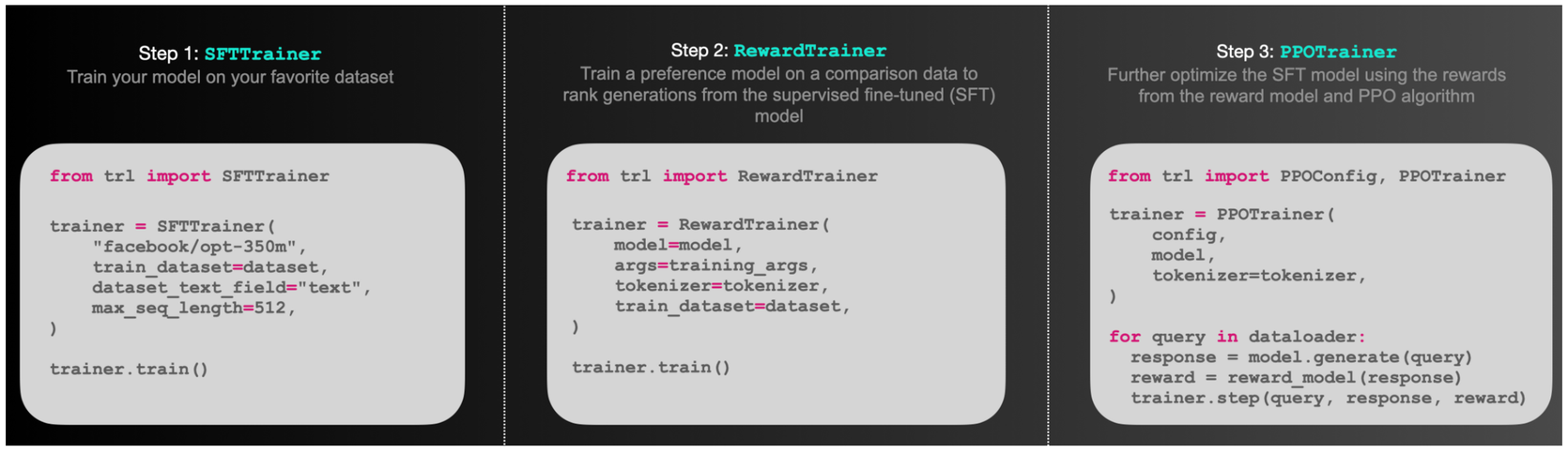
  - Replaces RM + RL by a simple classification loss:

$$\mathcal{L}_{\mathrm{DPO}}(p_\theta, p_{\mathrm{ref}}) = -\mathbb{E}_{(x,s^w,s^l)\sim D}\left[\log\sigma\left(\beta\log\frac{p_\theta(y^w|x)}{p_{\mathrm{ref}}(y^w|x)} - \beta\log\frac{p_\theta(y^l|x)}{p_{\mathrm{ref}}(y^l|x)}\right)\right]$$

  - Simple; removes difficult parts of the pipeline
  - Implementation quite easier to use !

In practice

- Look towards **Huggingface** (Models, datasets, tools .. for LLMs)
  - Proposes the *transformer* library
  - On top of it, *trl* - Transformer Reinforcement Learning



  - Notably, an example of a DPO pipeline for creating a 7B Llama mode

Alignment of Language Models

- **Still many issues**:
  - The biggest: **hallucinations**
    - Models trained to produce outputs that follow human preferences: ≠ truth !
  - Previous problems (privacy, bias, etc..) still exist

- **Current research directions:**
  - Improving alignment: with other methods/in other places
  - Improving fine-tuning: make it faster/more efficient
  - Facilitate model adaptation to new data/tasks
  - Couple models with Knowledge bases
    - Use retrieval models
  - **Next challenges**: legal (un-learning data), guarantees for trusted use ?