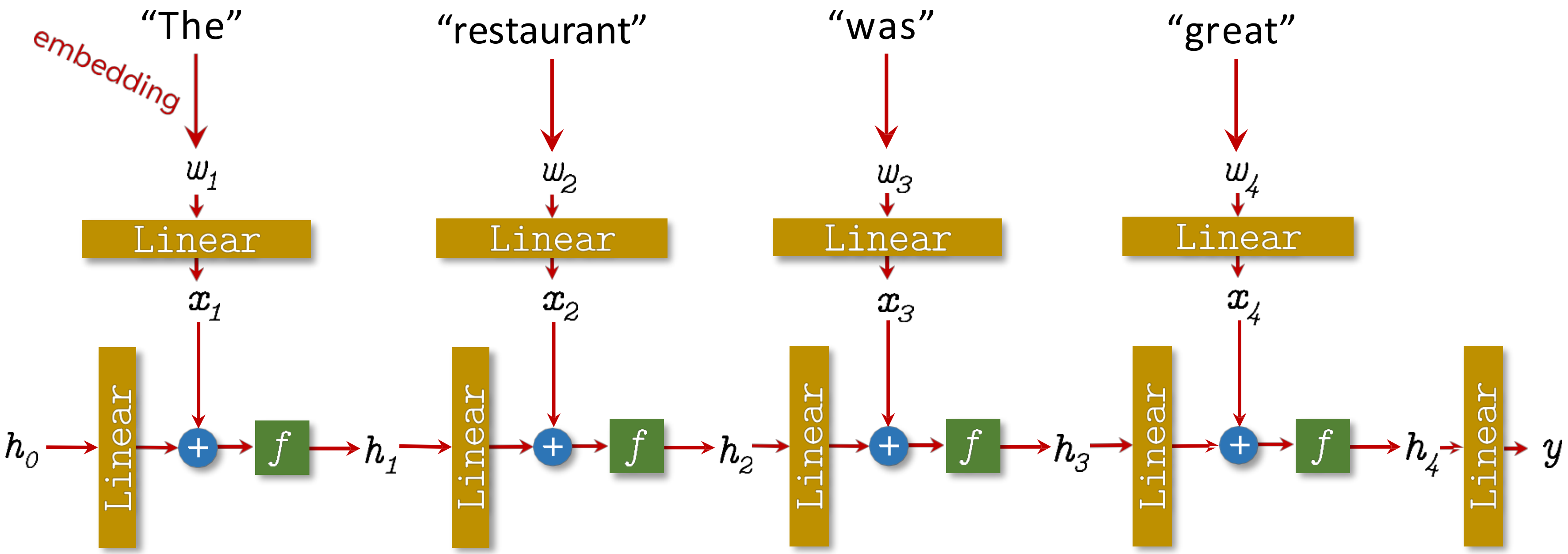
# CS 490: NATURAL LANGUAGE PROCESSING

Dan Goldwasser, Abulhair Saparov

Lecture 8: Attention and Transformers

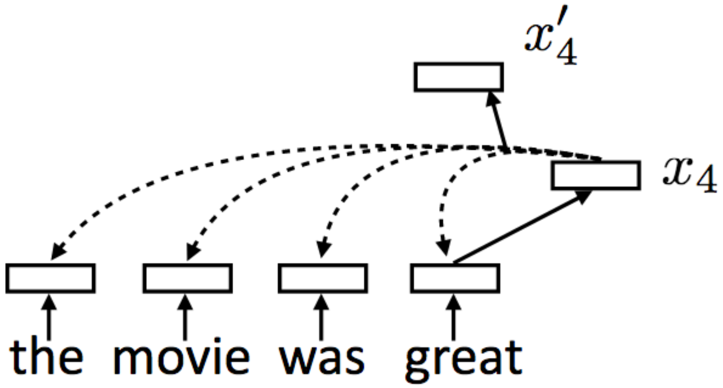
# RECAP FROM PREVIOUS LECTURE

- Models the sequential (word-by-word) processing nature of human language processing.



# Self-attention

- A way to represent structure
  - Each word forms a query which computes attention over each word



$$\alpha_{i,j} = \mathrm{softmax}(x_i^\top x_j)$$ scalar

$$x'_i = \sum_{j=1}^{n} \alpha_{i,j} x_j$$ vector = sum of scalar * vector

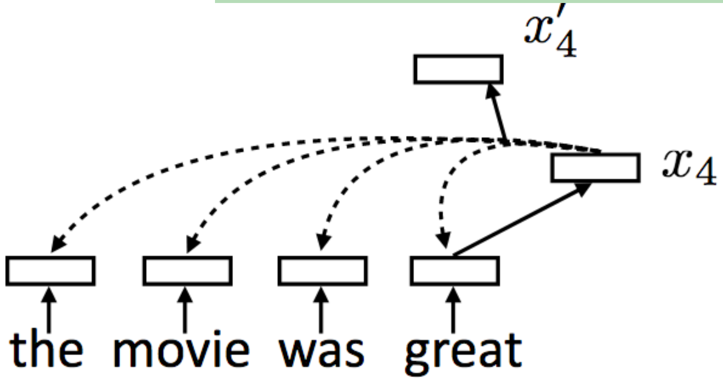
Vaswani et al. (2017)

# Self-attention

- A new way to represent structure
  - Each word forms a query which computes attention over each word

Extension – allow multiple attention heads

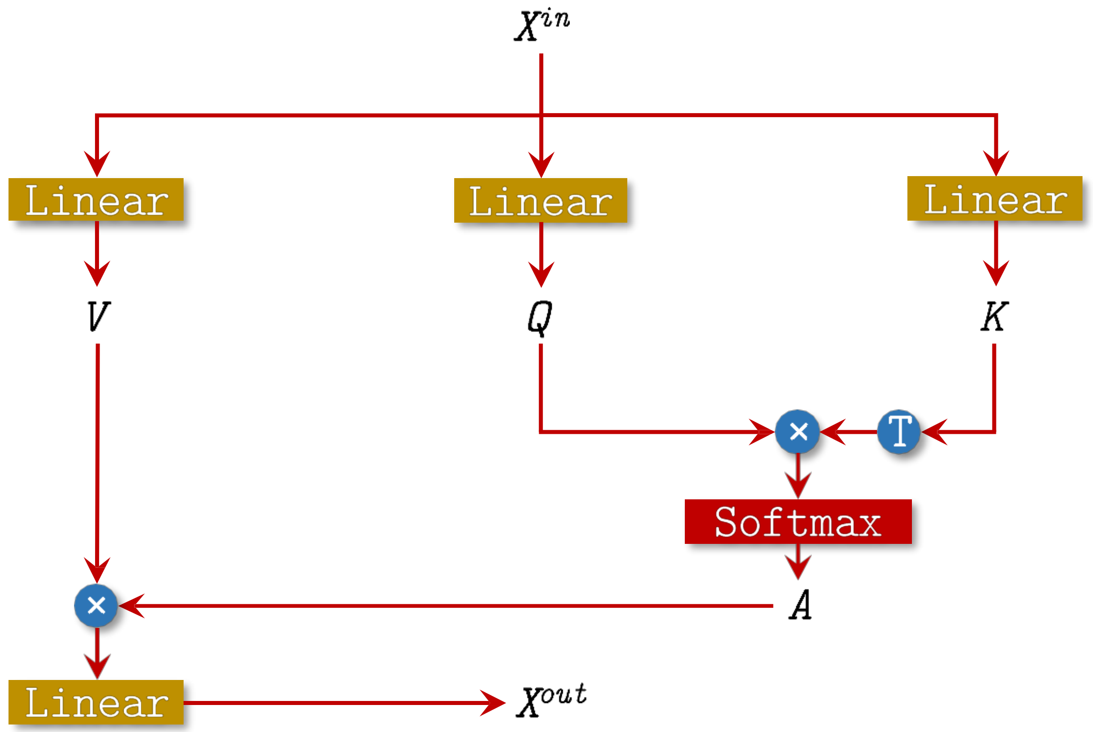
$$\alpha_{k,i,j} = \mathrm{softmax}(x_i^\top W_k x_j) \quad x'_{k,i} = \sum_{j=1}^{n} \alpha_{k,i,j} V_k x_j$$



$$\alpha_{i,j} = \mathrm{softmax}(x_i^\top x_j)$$ scalar

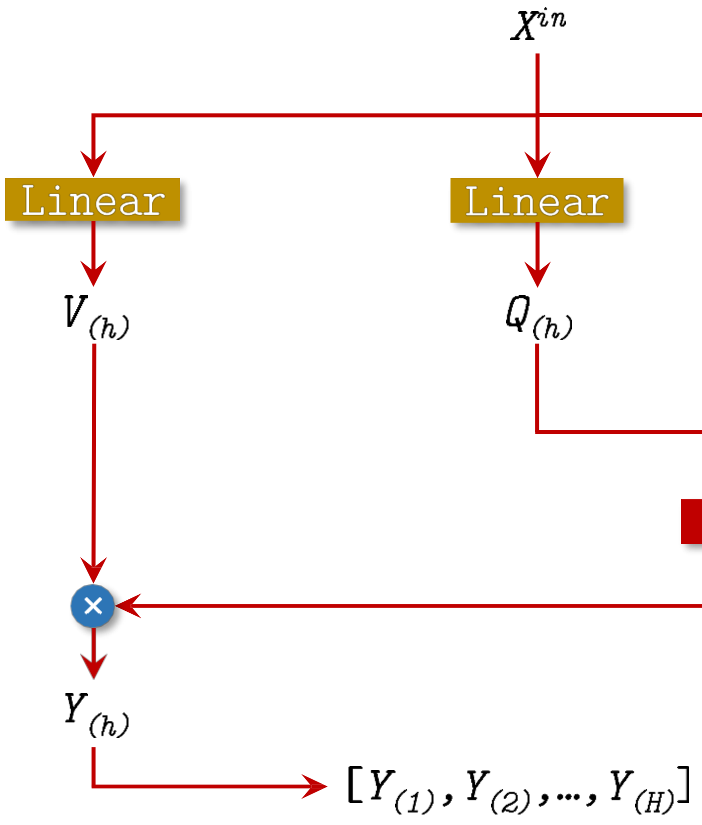
$$x'_i = \sum_{j=1}^{n} \alpha_{i,j} x_j$$ vector = sum of scalar * vector

# ATTENTION LAYER



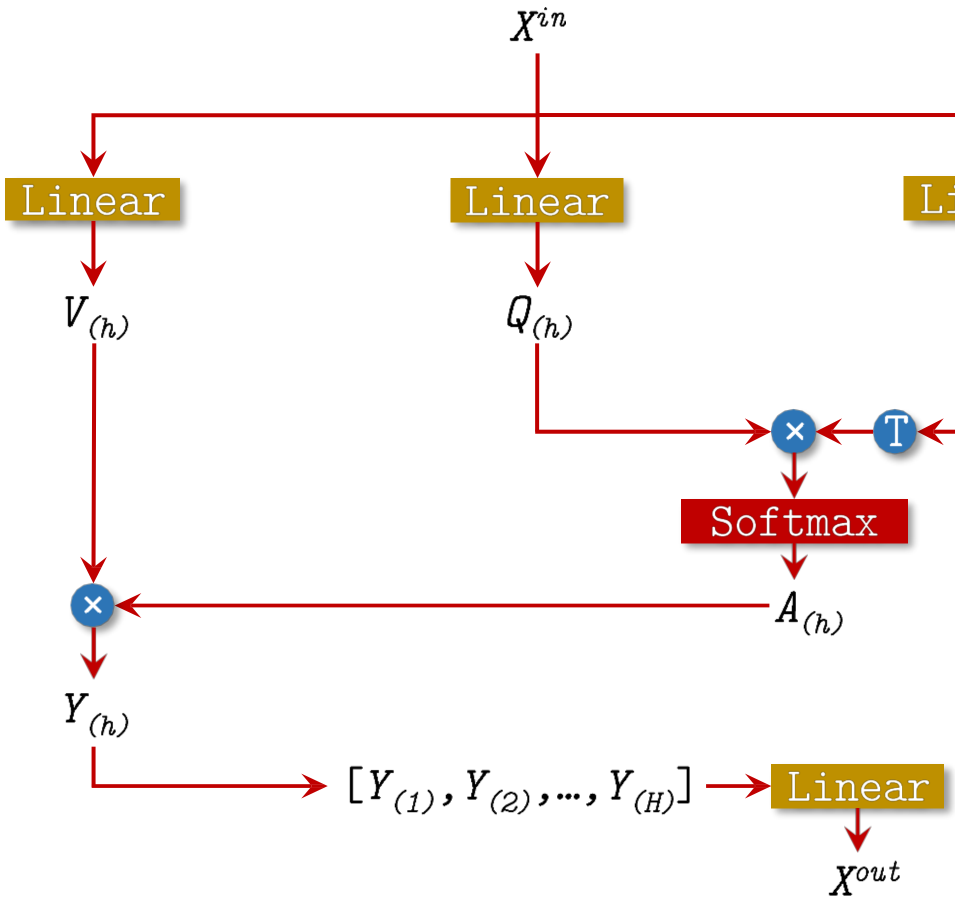
This is the output of the attention layer.
But recall that due to the residual connections, the actual output is $X^{in} + X^{out}$.

# MULTI-HEAD ATTENTION



Concatenate the outputs from each head into one large matrix.

# MULTI-HEAD ATTENTION



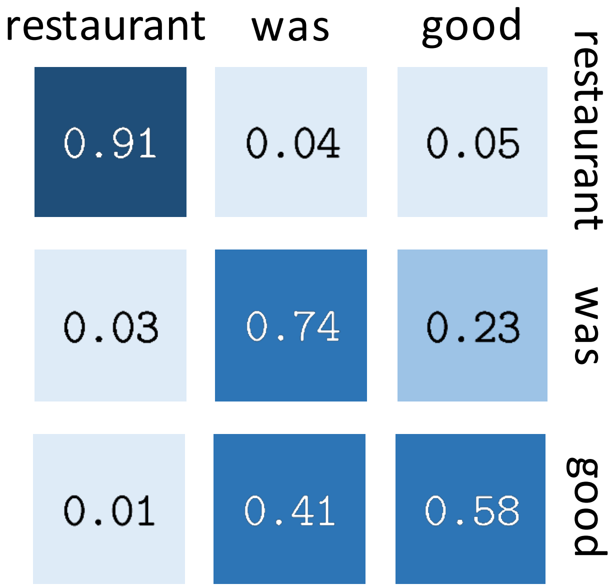
The resulting matrix is too large, since we need to add it back to $X^{in}$. It's dimension is $n \times Hd_{model}$. So we use a linear layer to resize it. Due to the residual connection, we add the output back to the input: $X^{in} + X^{out}$

# WHY MULTIPLE HEADS?

- Different attention heads can perform different computations.
- For example one attention head can compute syntactic relations:
  - “I run a small business” vs “I went for a run”
  - To compute the part-of-speech of “run”, it helps to attend the word immediately before: “I” vs “a”.
  - “a run” indicates that “run” is a noun.
  - “I run” indicates that “run” is a verb.
- A second attention head can compute semantic information:
  - What is the subject of the run?
  - In both examples above, the subject is “I”.

# CAUSAL MASK

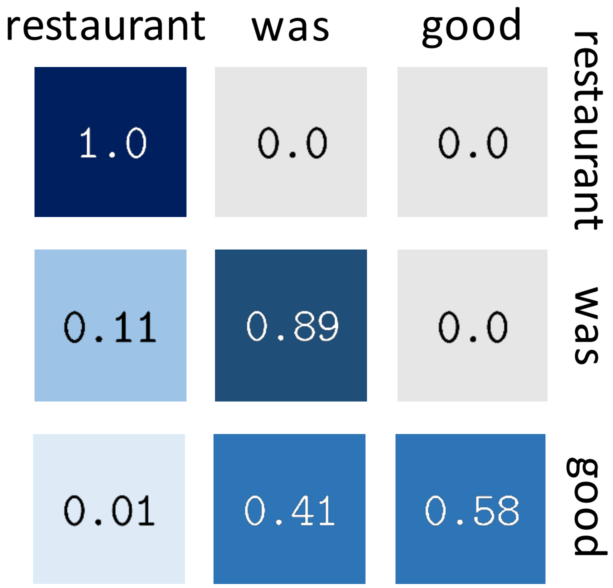
- Example of attention matrix without mask:



| | restaurant | was | good |
|---|---|---|---|
| restaurant | 0.91 | 0.04 | 0.05 |
| was | 0.03 | 0.74 | 0.23 |
| good | 0.01 | 0.41 | 0.58 |

# CAUSAL MASK

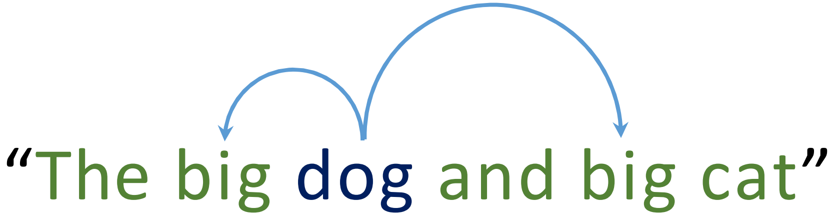
- Example of attention matrix with a causal mask:



| restaurant | was | good | |
|---|---|---|---|
| 1.0 | 0.0 | 0.0 | restaurant |
| 0.11 | 0.89 | 0.0 | was |
| 0.01 | 0.41 | 0.58 | good |

# POSITIONAL EMBEDDING

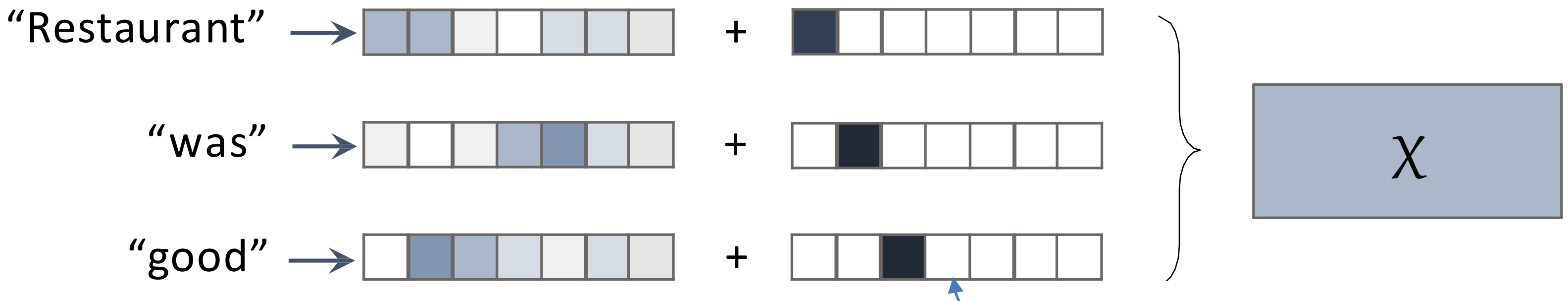
- Suppose the input $X$ to the attention layer has no position information (it only has word information).
- And suppose we have no causal mask.
- The attention layer is not able to compute the relative positions of tokens:
  - E.g. it can’t determine which token immediately follows any other token.
  - Suppose the word “dog” has high attention weight with “big”.
  - Since the embeddings of both “big” words are identical, the attention weight between dog and both big’s are the same.



“The big dog and big cat”

# POSITIONAL EMBEDDING

- Thus, position information is explicitly added to the embeddings:
- There are many kinds of positional embeddings.
- Token embeddings and positional embeddings can be summed, multiplied, or concatenated, etc.
  - Lots of ways to incorporate position information into embeddings.



$p_{i,t}$ is the i$^{th}$ element in the p$_i$ position vector

$$\begin{cases} p_{i,2t} = \sin(i/10000^{2t/d}) \\ p_{i,2t+1} = \cos(i/10000^{2t/d}) \end{cases}$$

# LAYER NORMALIZATION

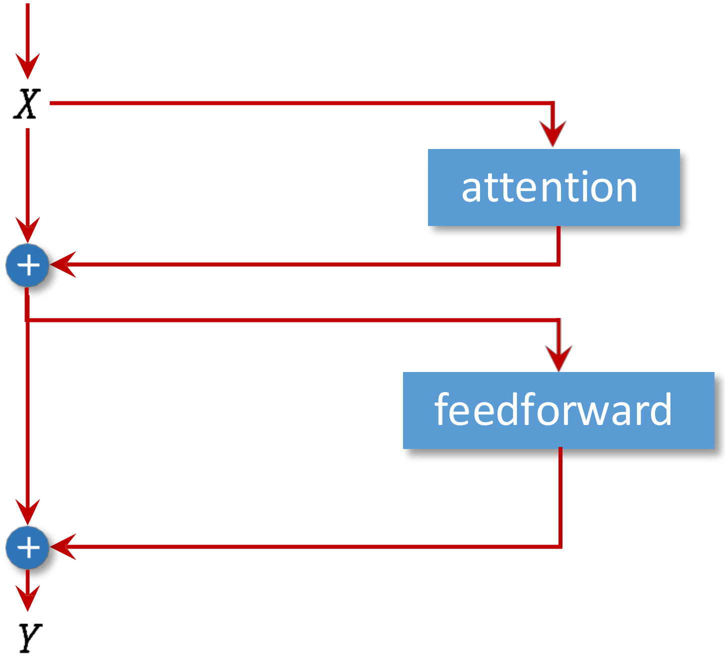
- Suppose we are training a transformer on a classification task, so we have a `softmax` operation at the end of the network.
- Also suppose the input embeddings have large magnitude,
- It’s very likely that the magnitude of the embeddings stays large throughout the transformer layers, up to the last softmax operation.
- Recall that the derivative of the softmax is close to zero if the input is a large positive or negative value.
  - Hint: The logistic function (i.e., sigmoid) is equivalent to softmax in two dimensions.
- Thus, in this example, the gradient would be very close to zero, and training would be extremely slow.

# LAYER NORMALIZATION

- Thus, transformers with many layers can also sometimes suffer from vanishing gradients.
- To avoid this, transformers use layer normalization.
  - Idea: Keep the activations close to 0 and not too negative or too positive.

# LAYER NORMALIZATION

“Restaurant was good”

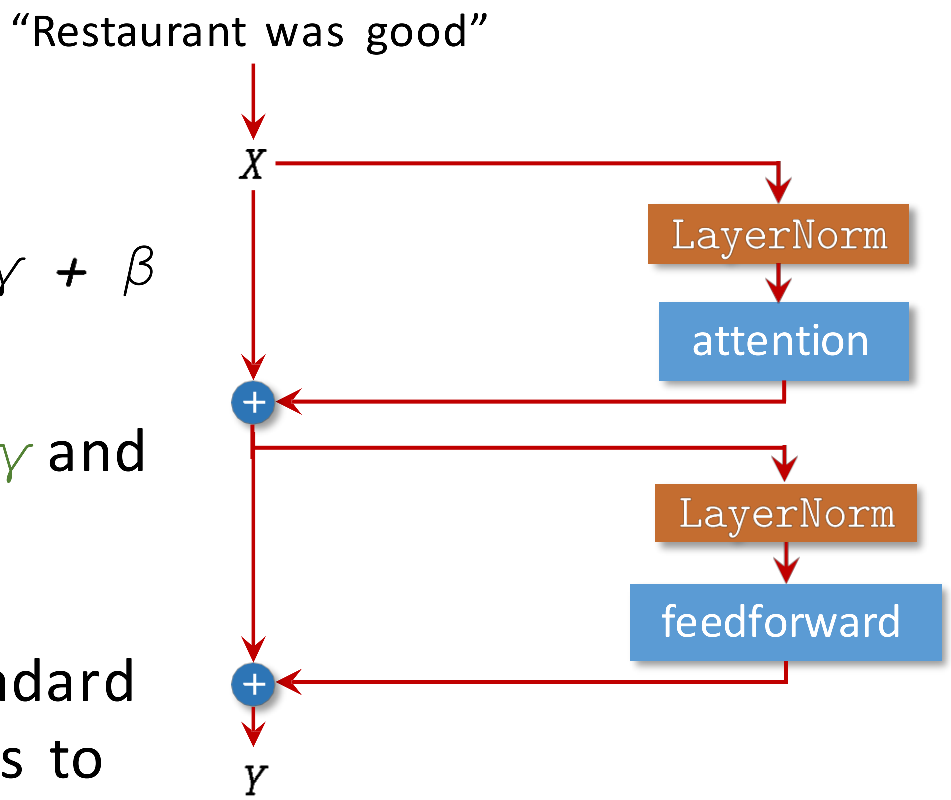


# LAYER NORMALIZATION

$$\mathtt{LayerNorm}(x_i) = \frac{x_i - avg(x_i)}{\sqrt{var(x_i) + \varepsilon}} \circ \gamma + \beta$$

Where $\varepsilon$ is a small fixed constant, $\gamma$ and $\beta$ are vectors of learnable weights.

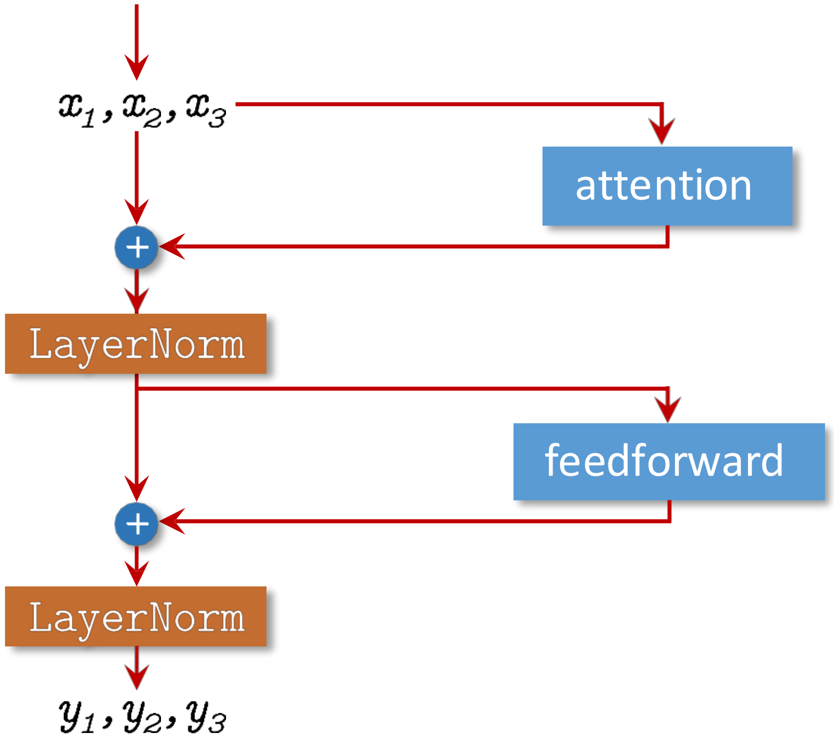
Since we scale the input by its standard deviation, layer normalization helps to prevent the activations from attaining very large positive or negative values.



# POST-LAYER NORMALIZATION

The original transformer paper used post-layer normalization.

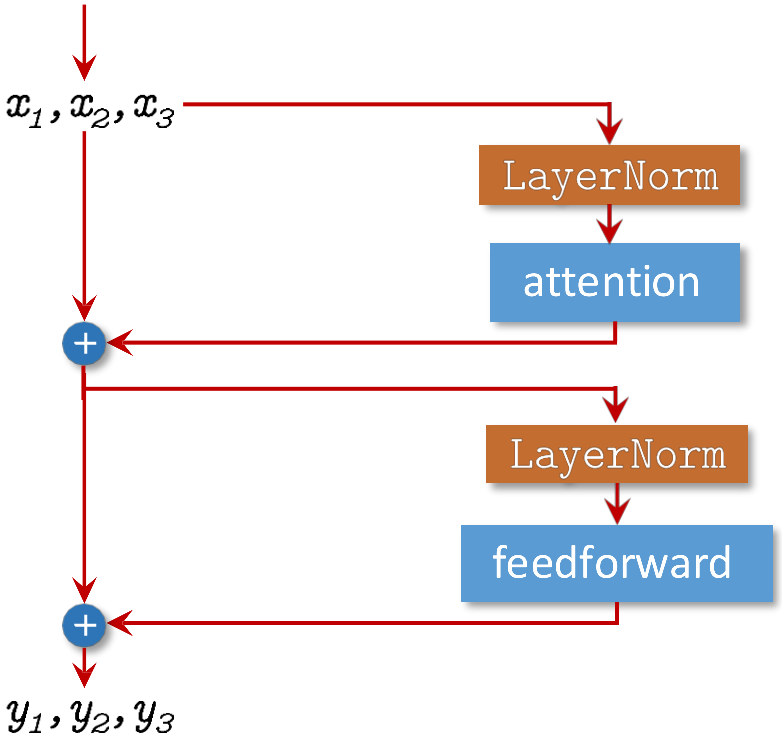
Layer normalization was applied on the residual stream (i.e., *after* residual connection).



[Vaswani et al., 2017]

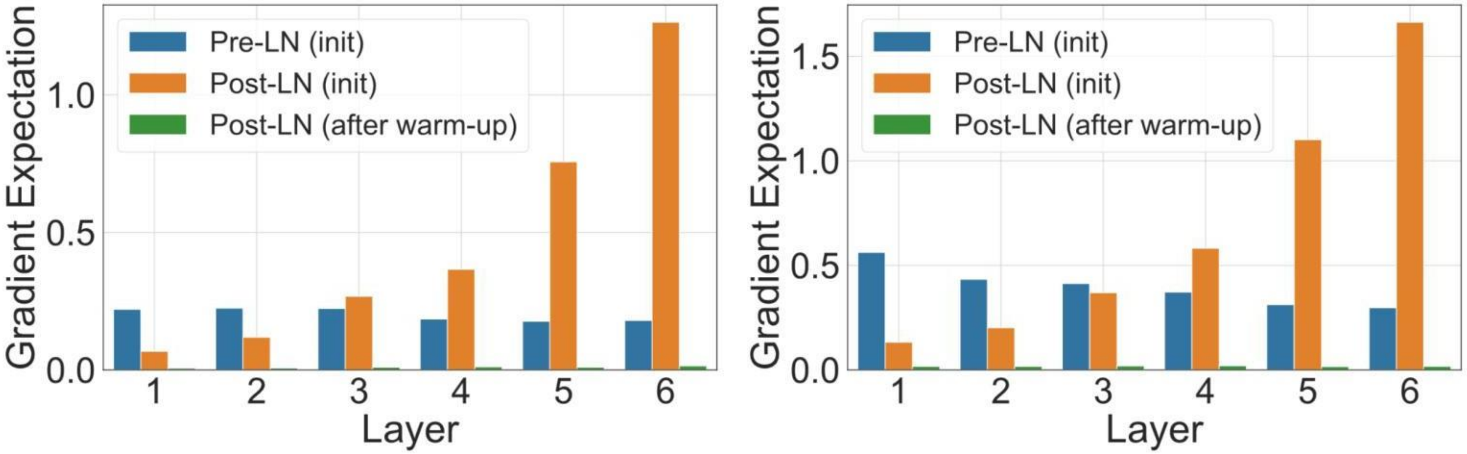
# PRE-LAYER NORMALIZATION

Xiong et al., 2020, proposed moving the layer normalization *before* the attention and FF blocks.



# PRE-LAYER NORMALIZATION

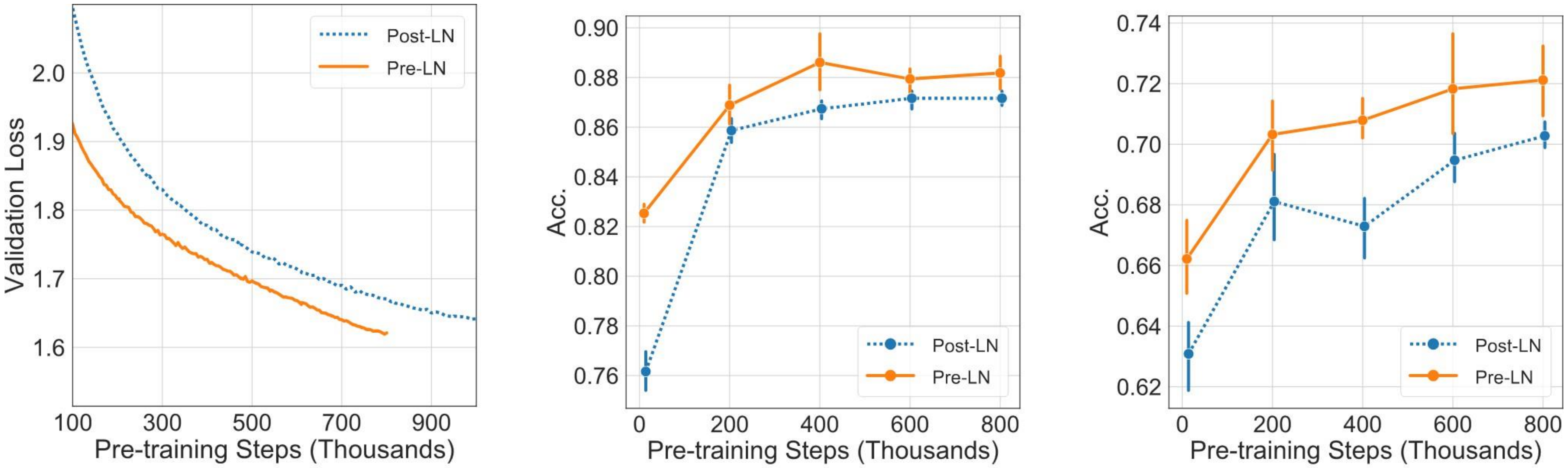
- Xiong et al., 2020, showed that the magnitudes of the gradients are more uniform across layers when using pre-layer normalization.
- Hypothesis: Learning occurs at a more uniform rate across layers when using pre-layer norm.



(a) $W^1$ in the FFN sub-layers (b) $W^2$ in the FFN sub-layers

# PRE-LAYER NORMALIZATION

- Measure empirical performance on masked language modeling, semantic similarity (Microsoft Research Paragraph Corpus), and textual entailment (Recognizing Textual Entailment dataset).



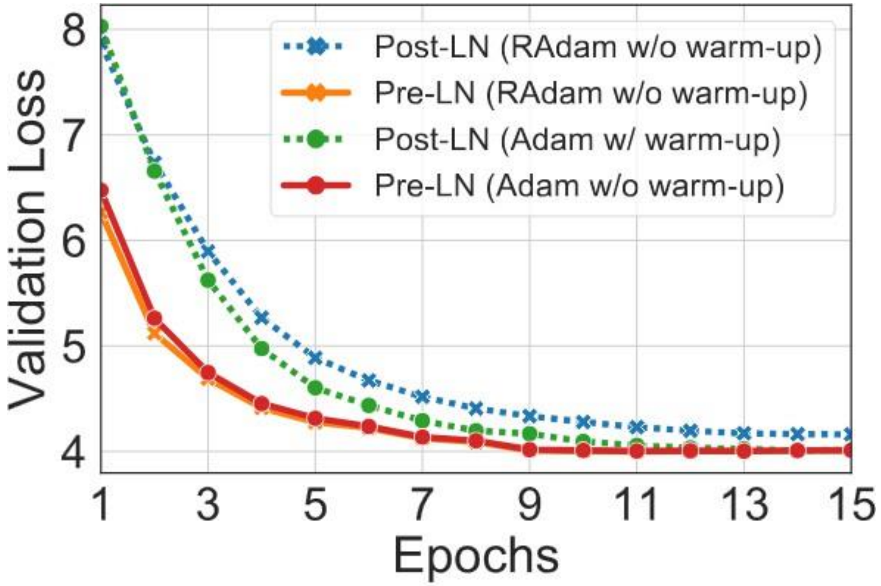
(a) Validation Loss on BERT

(b) Accuracy on MRPC

(c) Accuracy on RTE

# PRE-LAYER NORMALIZATION

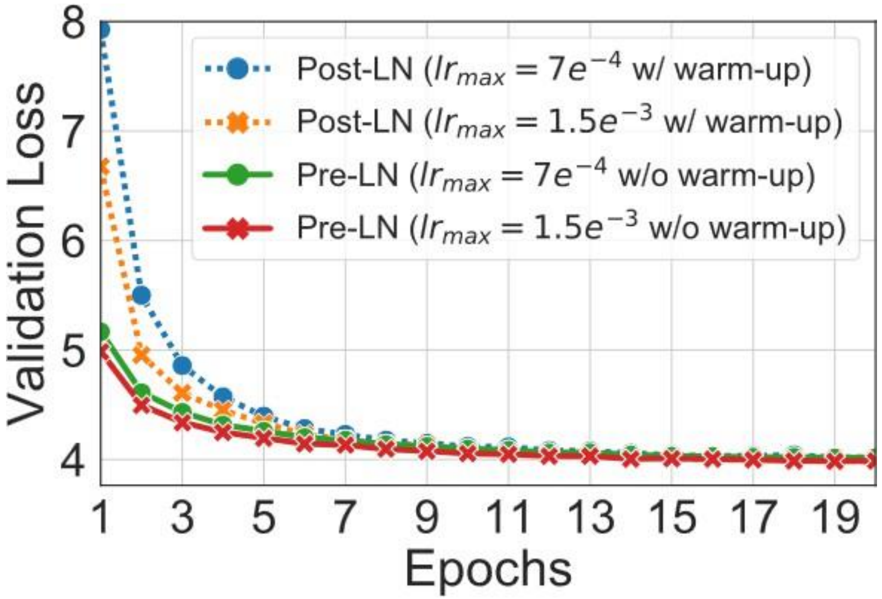
- Measure empirical performance on machine translation.



(a) Validation Loss (IWSLT)

# PRE-LAYER NORMALIZATION

- Measure empirical performance on machine translation.



(c) Validation Loss (WMT)

# RMS NORMALIZATION

- Zhang and Sennrich, 2019, proposed a simpler alternative to layer normalization, called root mean square layer normalization, or RMSNorm.

$$\mathrm{LayerNorm}(x_i) = \frac{x_i - avg(x_i)}{\sqrt{var(x_i) + \varepsilon}} \circ \gamma + \beta$$

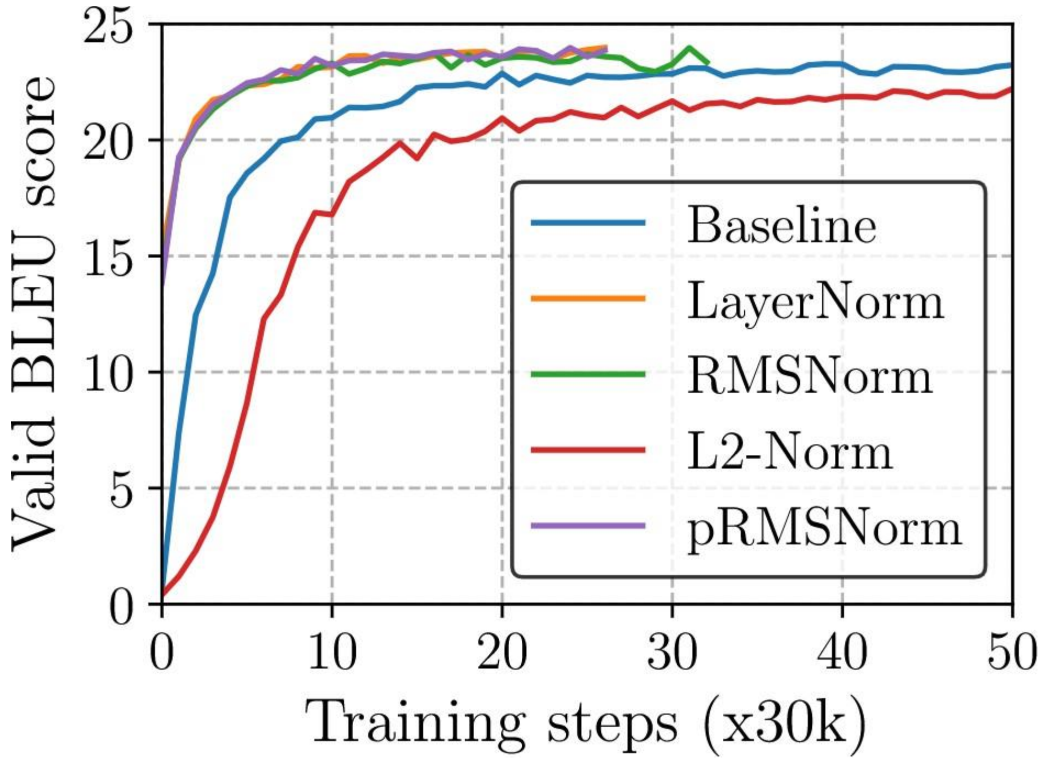
where $\varepsilon$ is a small fixed constant, $\gamma$ and $\beta$ are vectors of learnable weights.

$$\mathrm{RMSNorm}(x_i) = \frac{x_i}{RMS(x_i)} \circ \gamma \text{ where } RMS(x_i) = \sqrt{\frac{1}{n}\sum_{j=1}^{n} x_{i,j}^2}$$

- Note that RMSNorm is faster to compute.

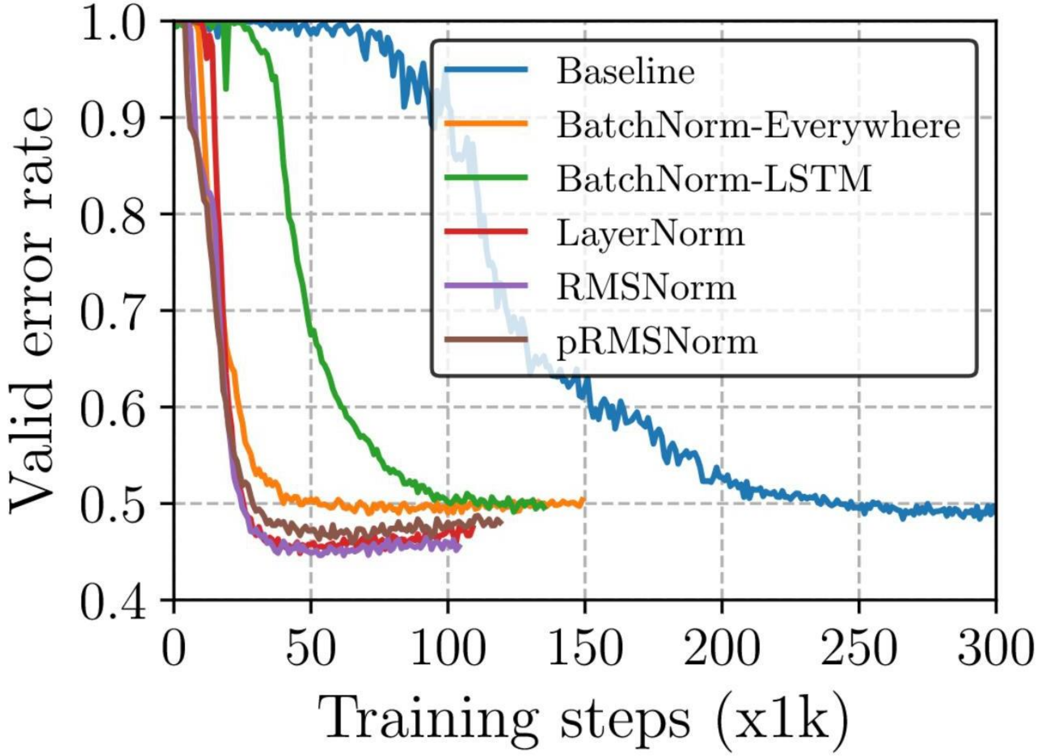
# RMS NORMALIZATION

- Measure empirical performance on machine translation.



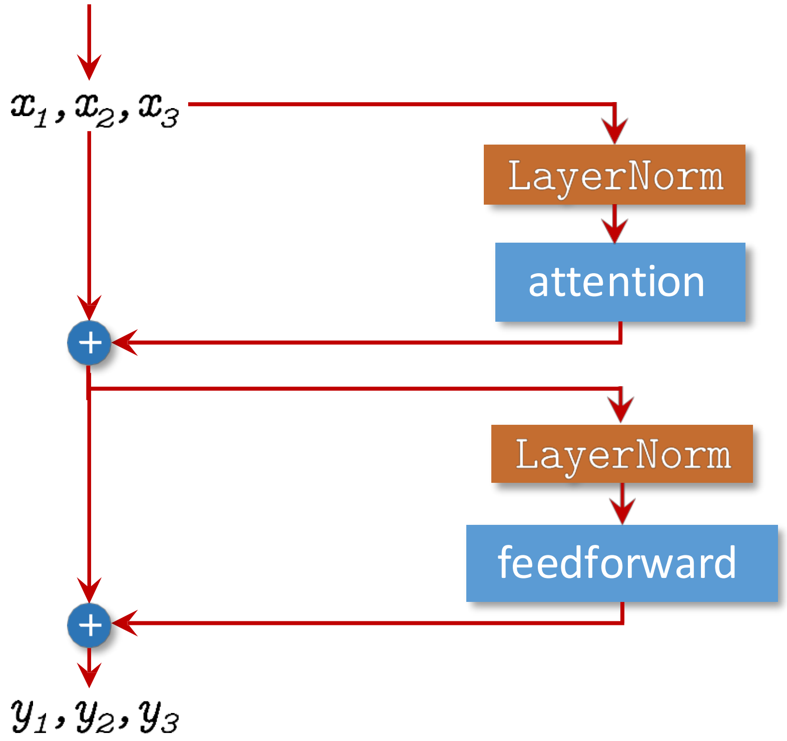
# RMS NORMALIZATION

- Measure empirical performance on question answering.



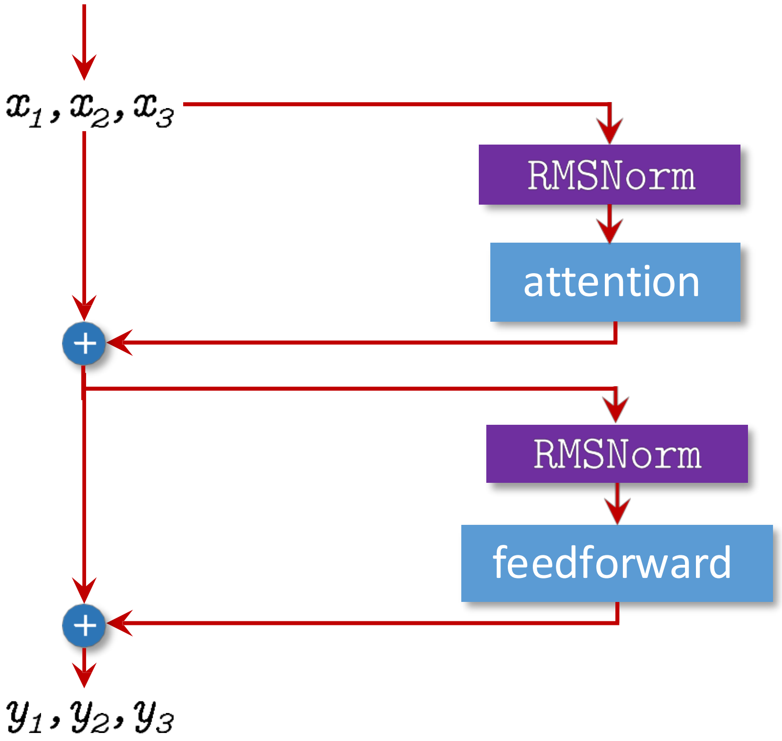
# PRE-LAYER NORMALIZATION AND RMSNORM

- Older transformer models (e.g., GPT-2) used pre-layer normalization.



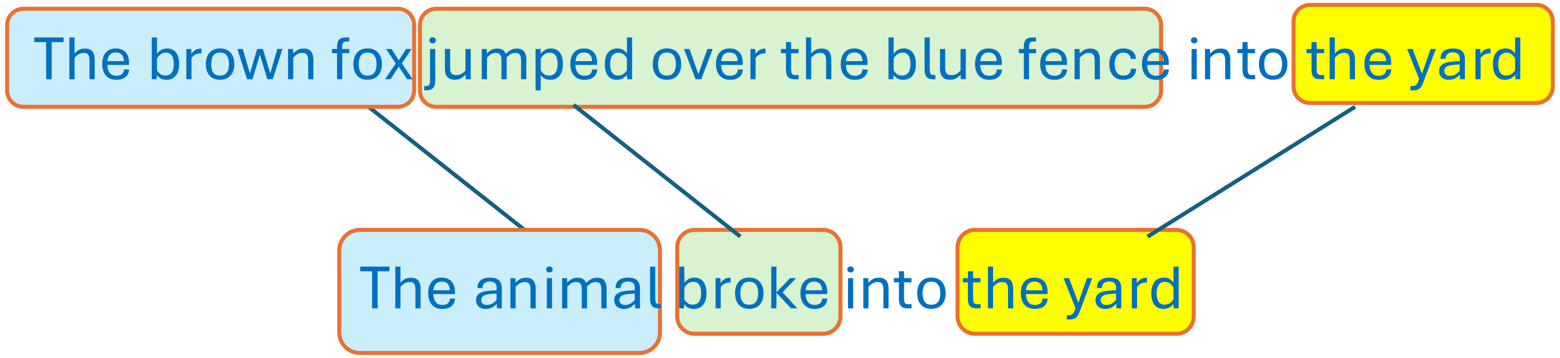
# PRE-LAYER NORMALIZATION AND RMSNORM

- Older transformer models (e.g., GPT-2) used pre-layer normalization.
- Recent models (e.g., Llama, DeepSeek, Qwen, Olmo) typically use pre-layer normalization with RMSNorm.



# Alignment-based problems

- Many problems in NLP require **alignment,** i.e., a mapping between text segments.
- **Alignment over inputs:** *Entailment* for example



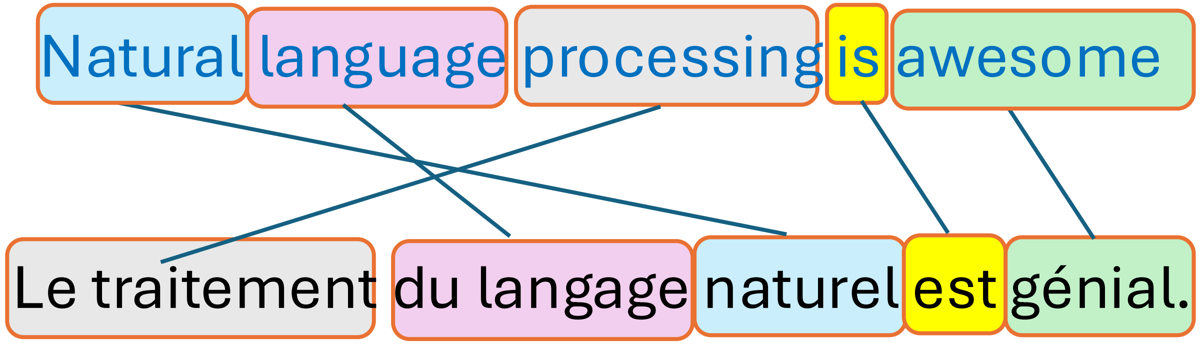
The brown fox jumped over the blue fence into the yard

The animal broke into the yard

**TE Task: does the premise entail the hypothesis?**

# Alignment-based problems

- Many problems in NLP require **alignment,** i.e., a mapping between text segments.
- **Alignment over output:** *automatic translation* for example



# Alignment-based problems

- In these problems the mapping from input to output is incomplete.
- They require an ”extra” step, the alignment serves as an explanation/justification for the output decision.
- E.g., *we can identify why every word in the translate text is needed by mapping it to the source sentence*.
- **Traditionally, this mapping was done explicitly.**

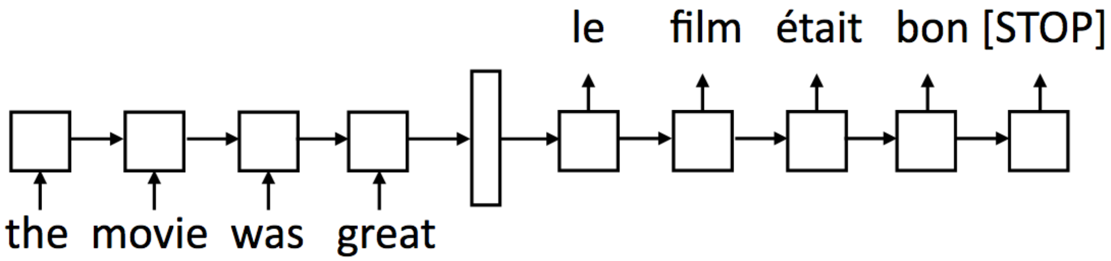
# *Attention based "reading" and "writing"*

- **The attention mechanism** and the transformer architecture can help in such tasks.
- The attention mechanism acts as an alternative to explicit alignments  and can be interpreted as soft alignments.

- Early works adapt an RNN based encoder-decoder approach to an attention based architecture.

# Encoder-Decoder Architecture

- Popular choice for NLP applications that require generation
  - NMT, summary, semantic parsing,..
- Encode sequence into a fixed size vector



- Now use that vector to produce a series of tokens as output from a separate LSTM decode

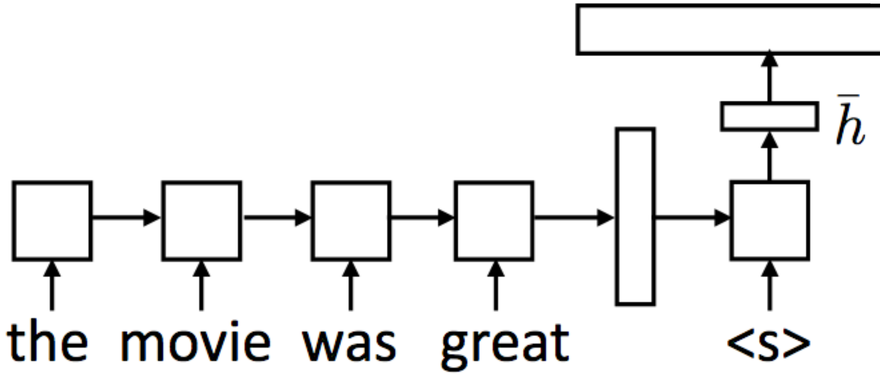
Sutskever et al. (2014)

# Encoder-Decoder Architecture

- Generate the next word, conditioned on previous words as well as hidden state

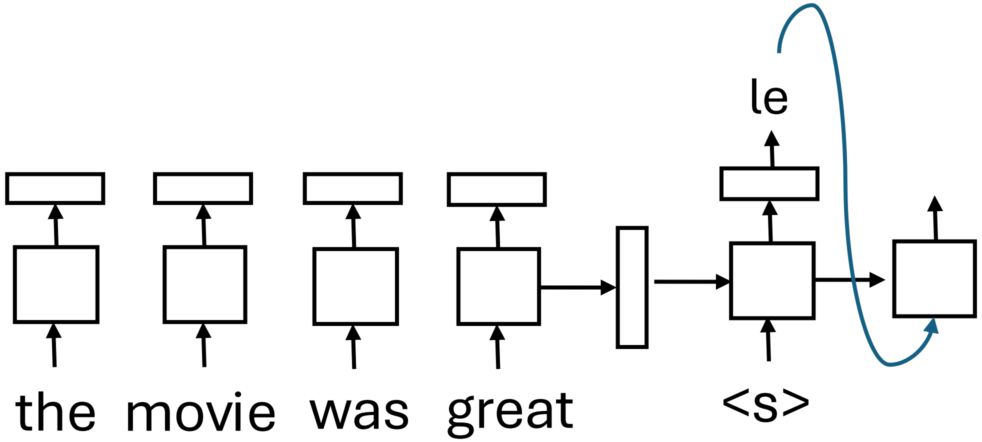
$$P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \mathrm{softmax}(W\bar{h})$$

$$P(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{n} P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1})$$



# Inference

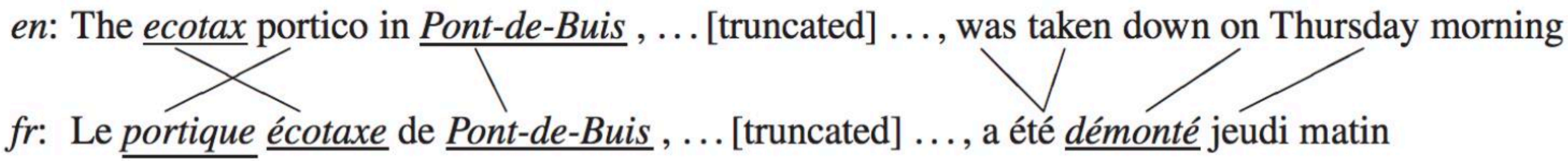
- At inference time, the predicted word can be used as input to the next word prediction



# Seq2Seq issues

- **Encoder-decoder models tend to repeat output sequences**
- Un garçon joue dans la neige ➔ A boy plays in the snow boy plays boy plays
  - **Some notion of coverage is needed**

- **Depending on context, some word should not be translated**



*en*: The *ecotax* portico in *Pont-de-Buis* , . . . [truncated] . . . , was taken down on Thursday morning

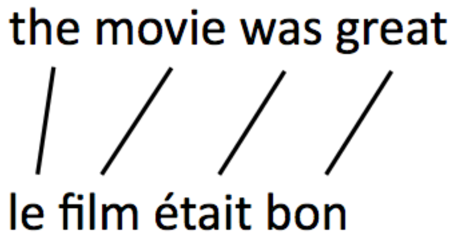
*fr*: Le *portique* *écotaxe* de *Pont-de-Buis* , . . . [truncated] . . . , a été *démonté* jeudi matin

- **Issues with long sentences, similar to LSTM generally**

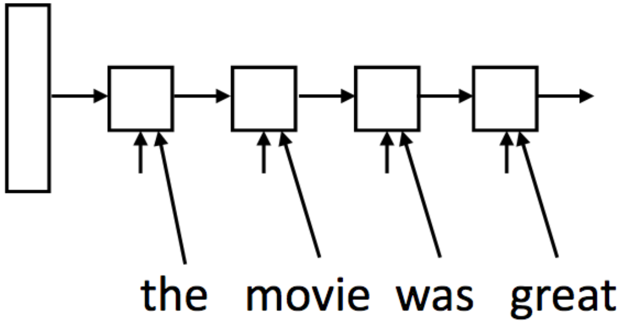
# Alignment

- **Recall out discussion about latent variables in structured prediction**
- Let’s assume the source and target sentences  can be mapped word-to-word

You can look at the corresponding word when translating, choice of output word still requires context



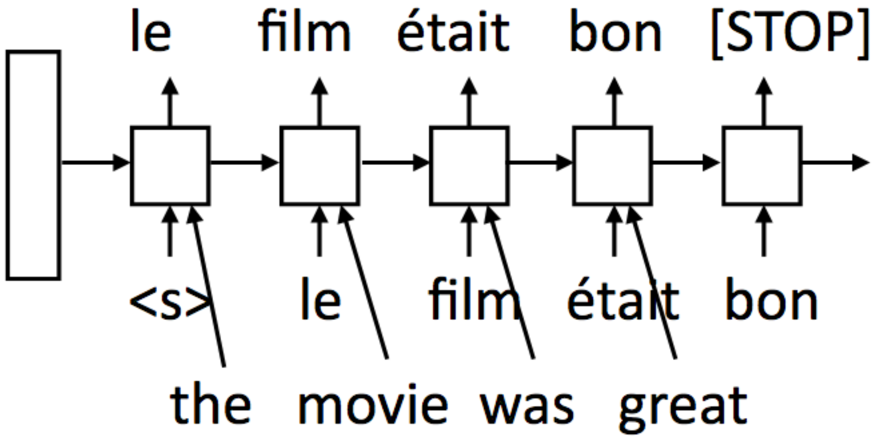
A lot of the “heavy lifting” is done by the mapping as opposed to the hidden state



# Alignment

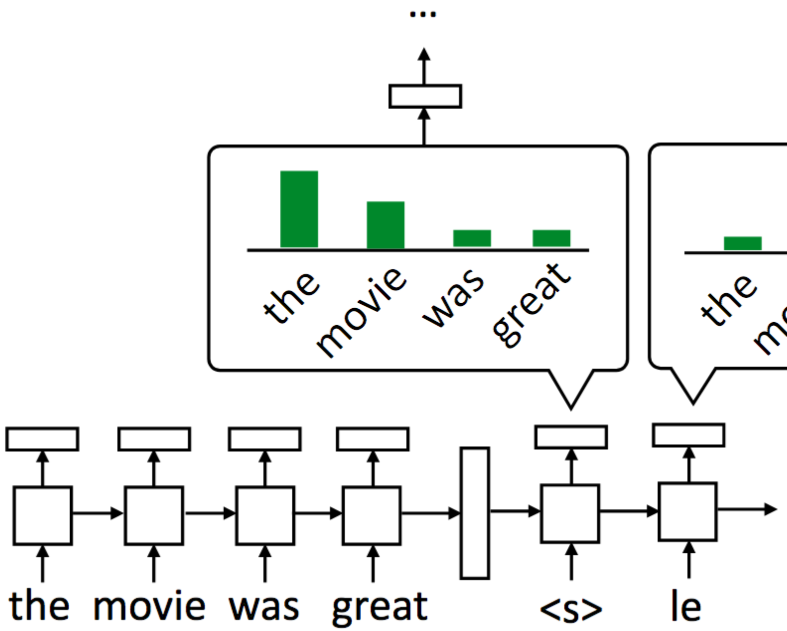
- **A lot of the "heavy lifting" is done by the mapping as opposed to the hidden state**
- Manually annotated alignments (hard to do)
- Alignments as latent variables (requires inference)
- **Soft alignments as part of the neural net**

No hard-coded alignments!



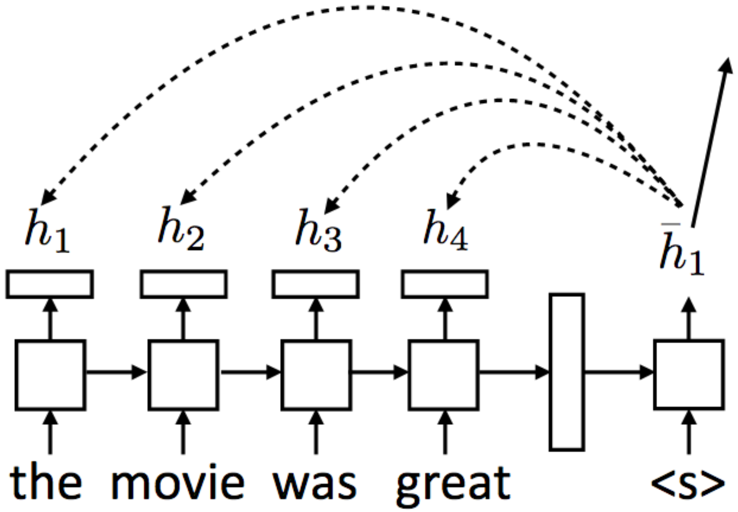
# Attention

- **Attention**: at each decoder state compute a distribution over the source inputs based on the current decoder state



# Attention

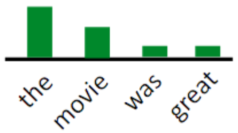
- For each decoder state compute the weighted sum of input states



$$c_i = \sum_j \alpha_{ij} h_j$$

- Weighted sum of input hidden states (vector)

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

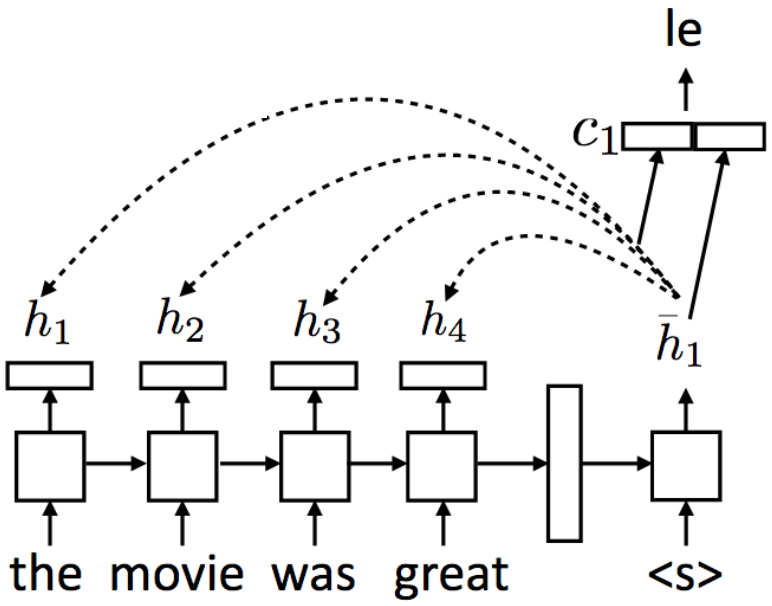


$$e_{ij} = f(\bar{h}_i, h_j)$$

- Unnormalized scalar weight

# Attention

- For each decoder state compute the weighted sum of input states

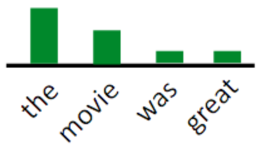


$$P(y_i|\mathbf{x}, y_1, \ldots, y_{i-1}) = \mathrm{softmax}(W[c_i; \bar{h}_i])$$

$$c_i = \sum_j \alpha_{ij} h_j$$

- Weighted sum of input hidden states (vector)

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})}$$

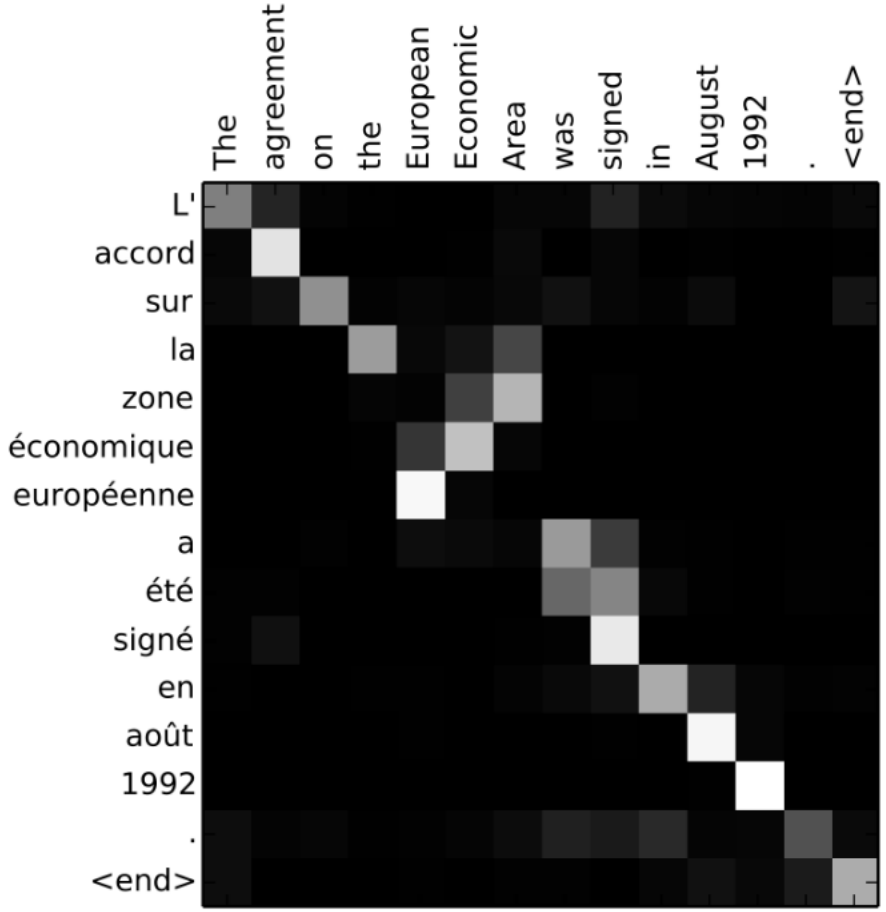


$$e_{ij} = f(\bar{h}_i, h_j)$$

- Unnormalized scalar weight

# Attention

- We can identify the source word context for output predictions



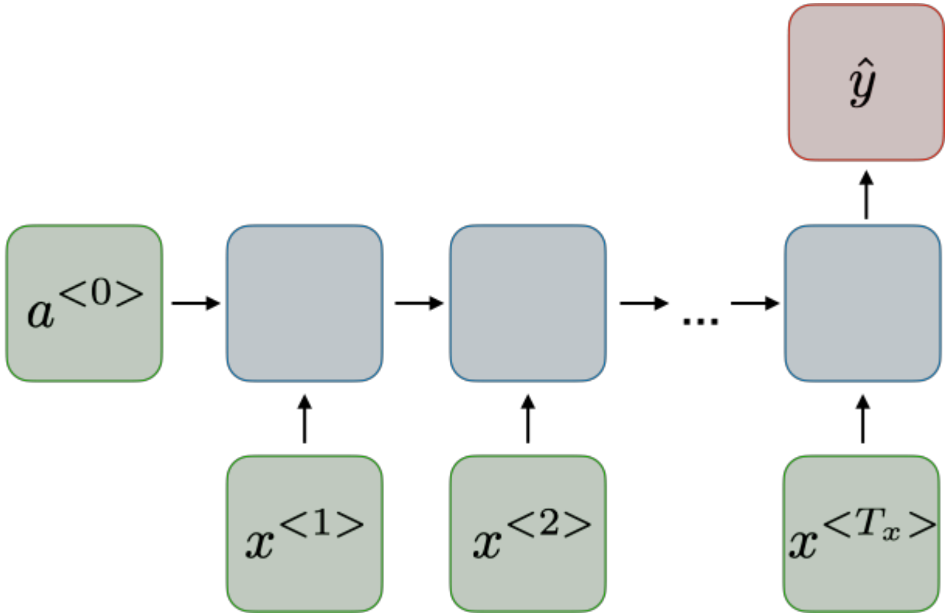
# Attention

- **Nice results for many applications:**
  - **Machine translation**: BLEU score of 14.0 on English-German -> 16.8 with attention, 19.0 with improved attention
  - **Summarization/headline generation**: bigram recall from 11% -> 15%
  - **Semantic parsing**: ~30% accuracy -> 70+% accuracy on Geoquery

Luong et al. (2015)Chopra et al. (2016)Jia and Liang (2016)

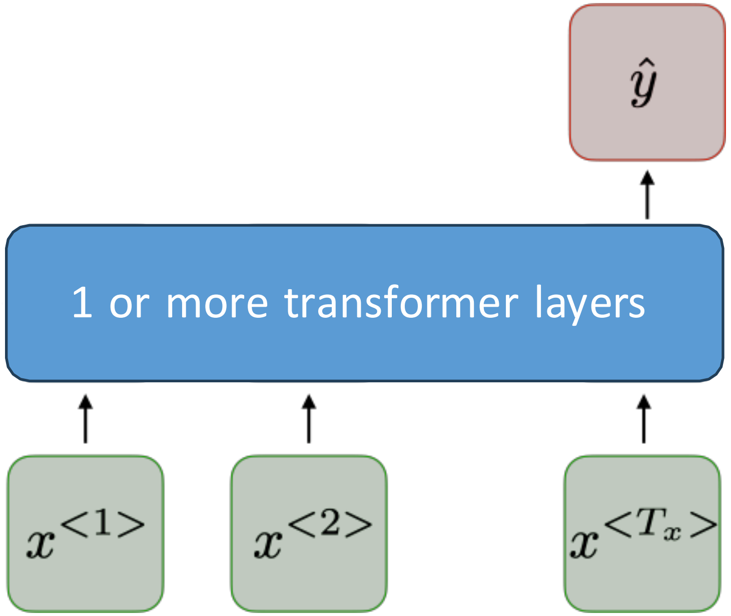
# TRANSFORMER APPLICATIONS

- Recall that RNNs can be used in a wide variety of architectures.



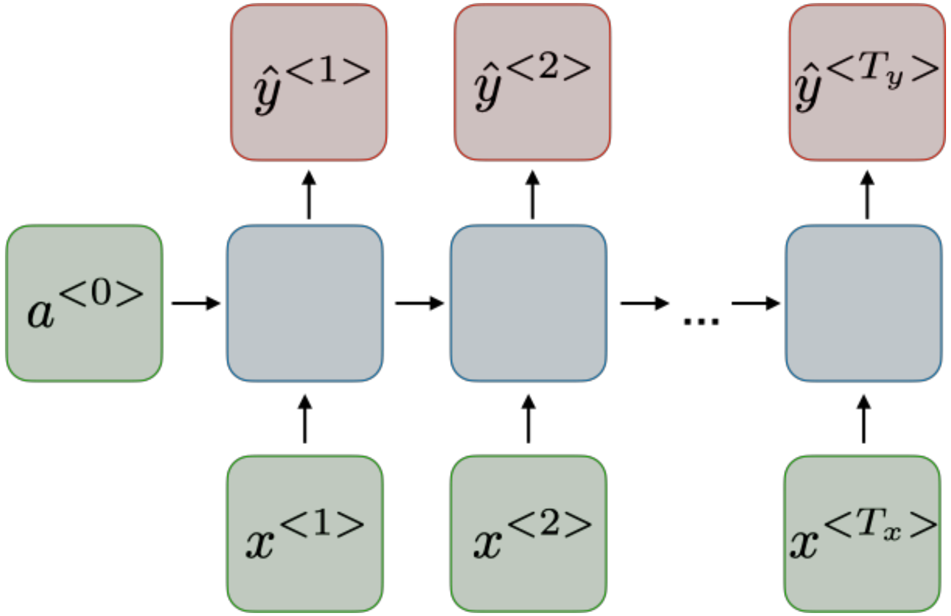
# TRANSFORMER APPLICATIONS

- Recall that RNNs can be used in a wide variety of architectures.
- Transformers can similarly be a component in lots of different models.



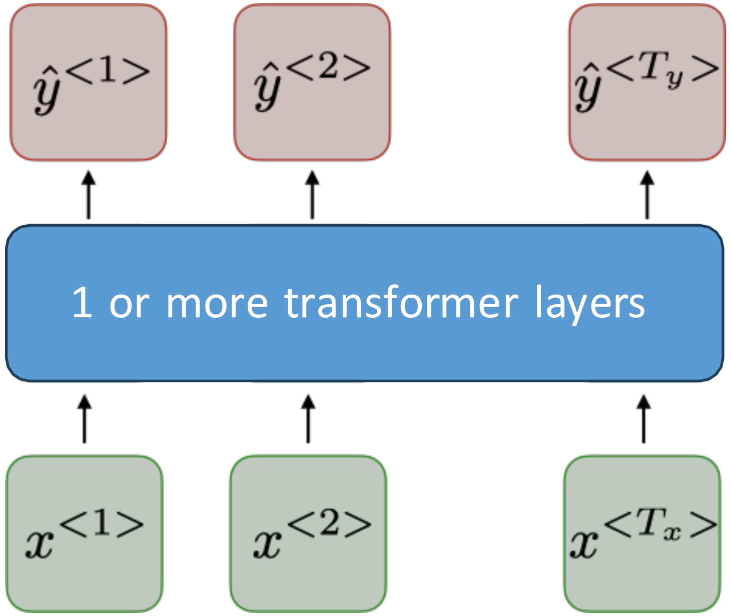
# TRANSFORMER APPLICATIONS

- Recall that RNNs can be used in a wide variety of architectures.
- Transformers can similarly be a component in lots of different models.



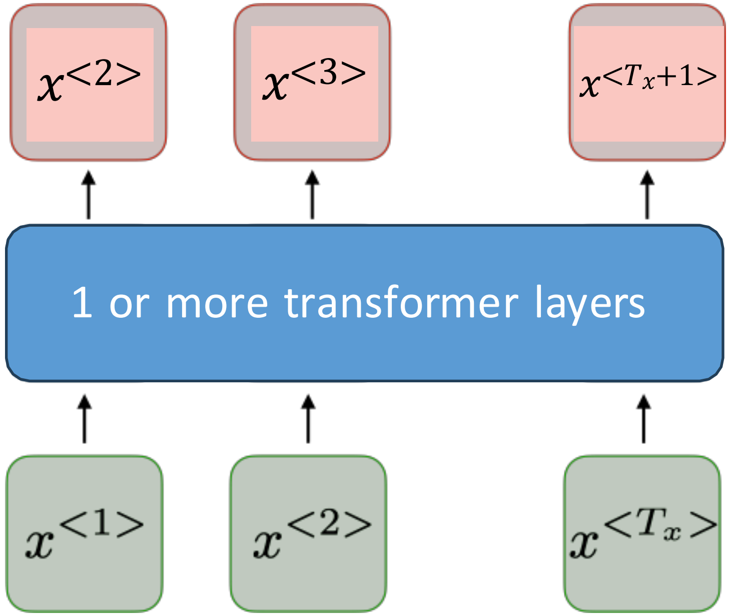
# TRANSFORMER APPLICATIONS

- Recall that RNNs can be used in a wide variety of architectures.
- Transformers can similarly be a component in lots of different models.



# TRANSFORMER APPLICATIONS

- Recall that RNNs can be used in a wide variety of architectures.
- Transformers can similarly be a component in lots of different models.



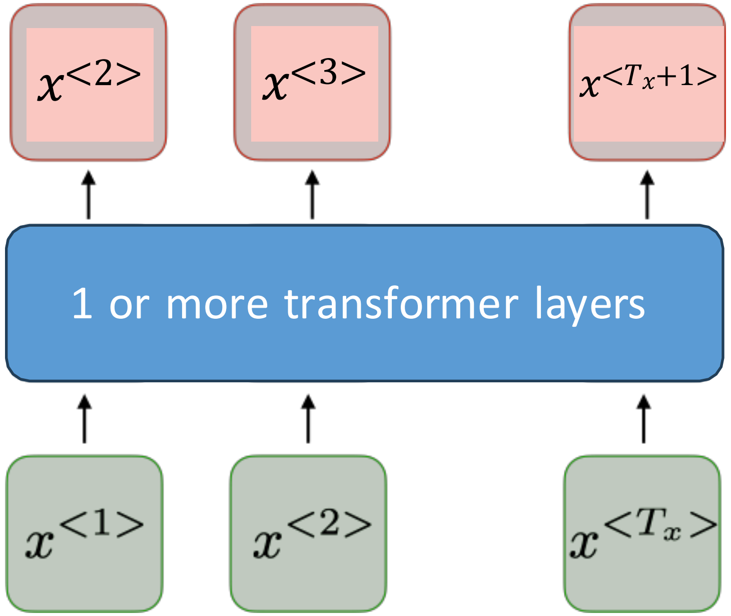
In the language modeling task, we often predict $n$ tokens, where the $i^{th}$ output token corresponds to the $i+1^{th}$ input token.

The loss is computed as the sum of the losses of all output tokens.

Often called autoregressive or causal language modeling.

# WHY THE CAUSAL MASK?

- Recall that RNNs can be used in a wide variety of architectures.
- Transformers can similarly be a component in lots of different models.



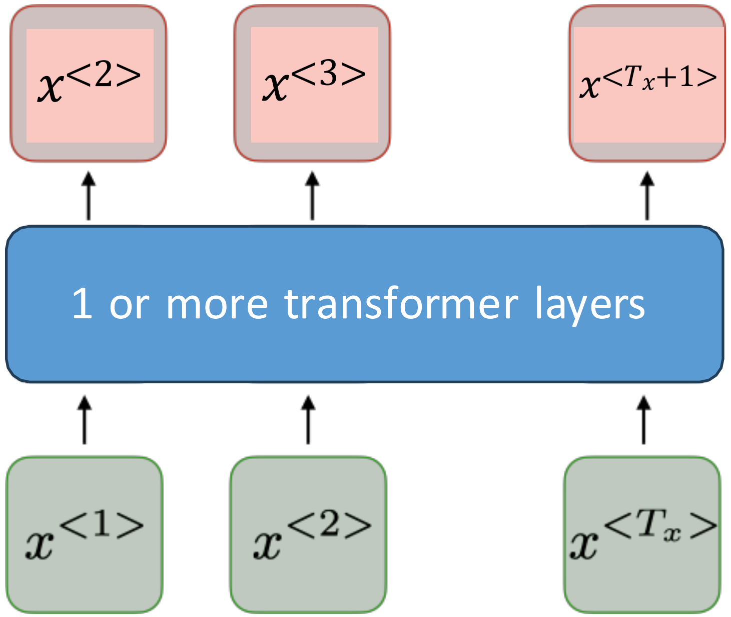
This motivates the causal mask.

We want $x^{<i>}$ to only attend to tokens that come before it.

Otherwise, it can simply cheat by copying $x^{<i+1>}$.

# WHY THE CAUSAL MASK?

- Recall that RNNs can be used in a wide variety of architectures.
- Transformers can similarly be a component in lots of different models.

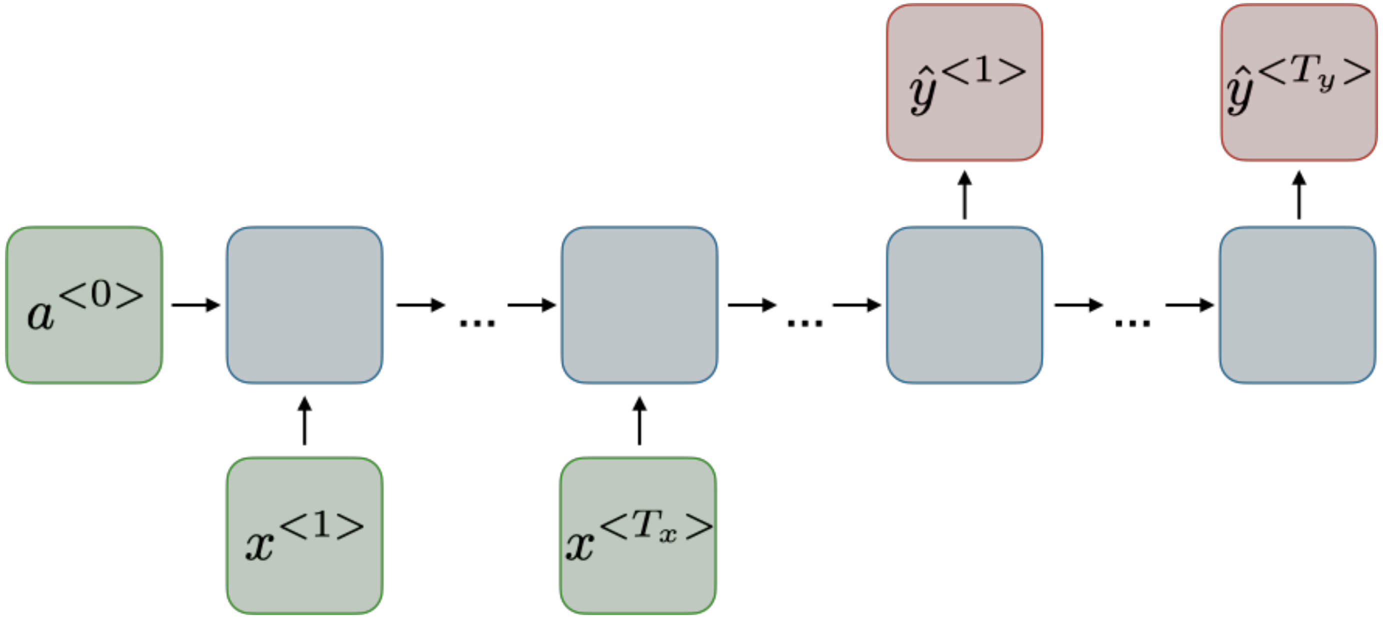


With or without the causal mask, the model still must predict the token after the last token.

The model can't "cheat" here by attending to future tokens.

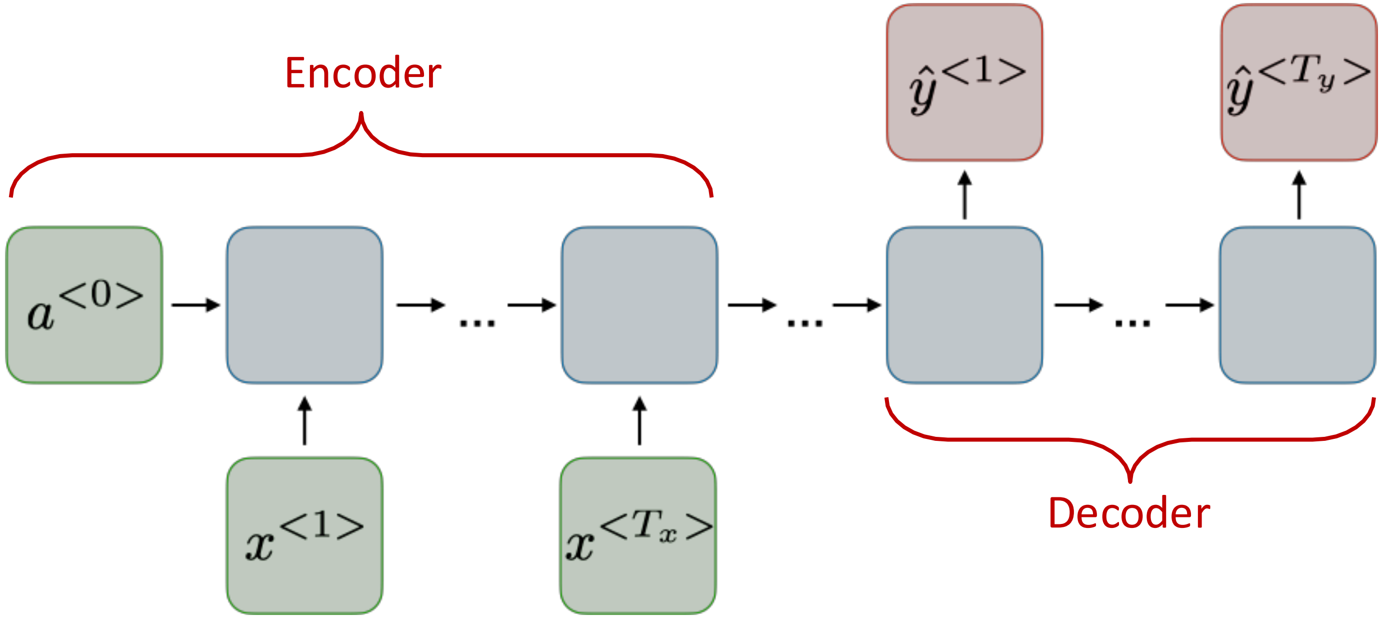
# TRANSFORMER APPLICATIONS

- Recall that RNNs can be used in a wide variety of architectures.
- Transformers can similarly be a component in lots of different models.

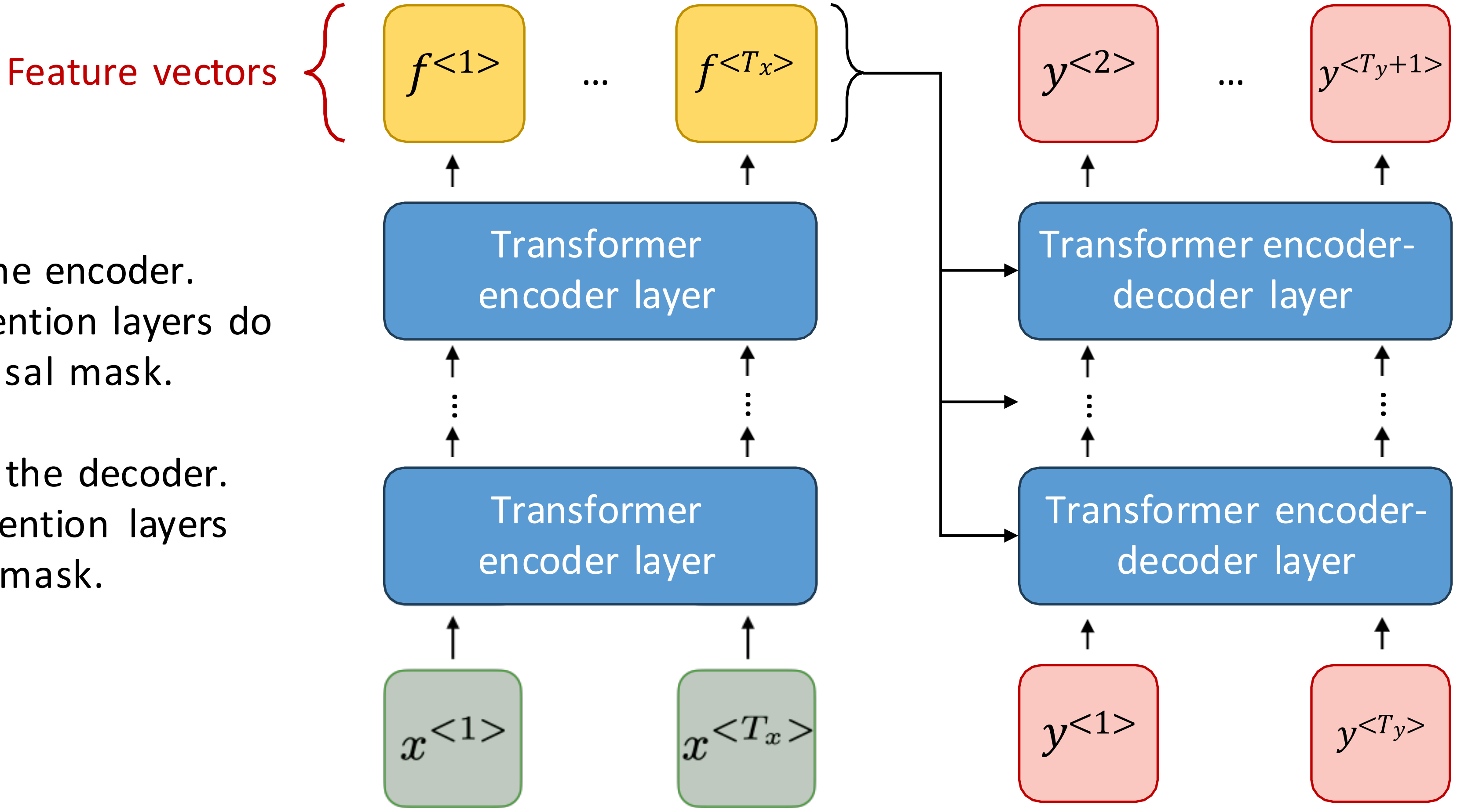


# TRANSFORMER APPLICATIONS

- Recall that RNNs can be used in a wide variety of architectures.
- Transformers can similarly be a component in lots of different models.



# TRANSFORMER ENCODER-DECODER

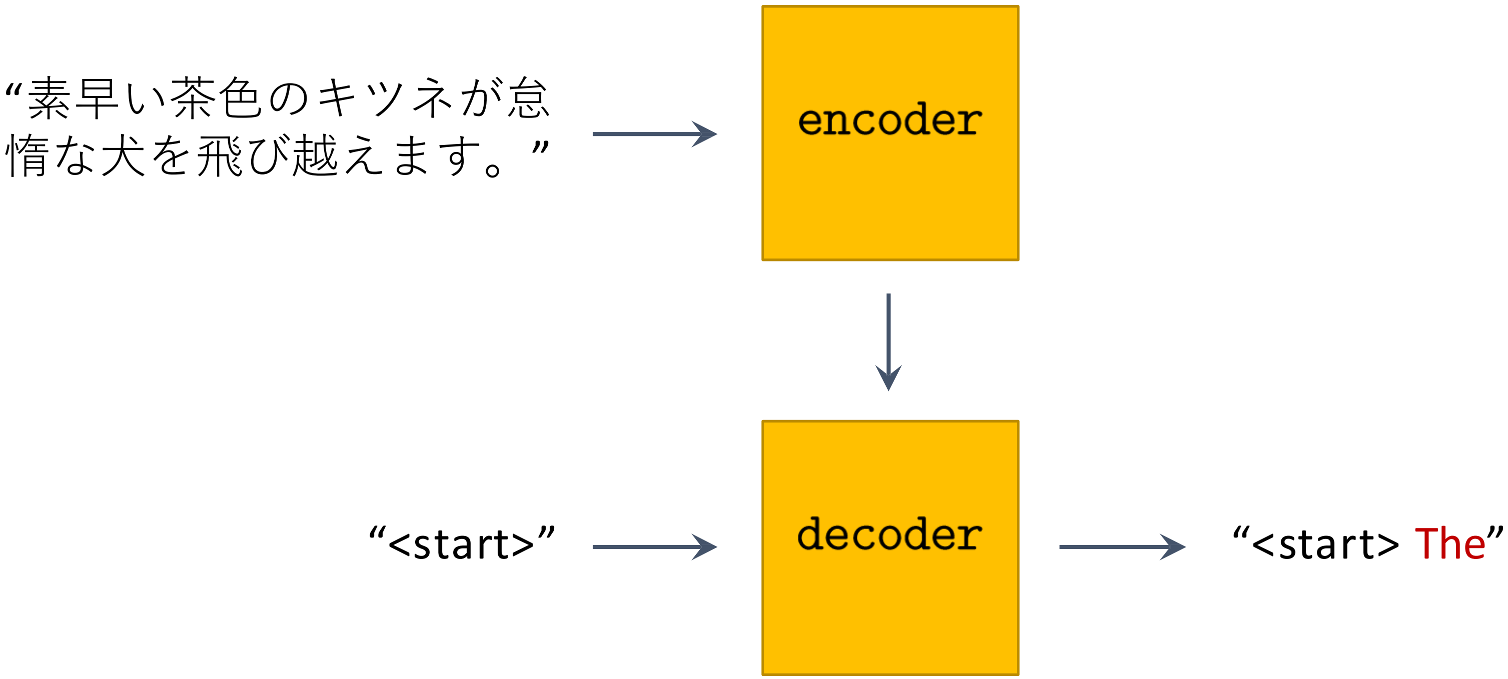


Feature vectors
$f^{<1>}$
...
$f^{<T_x>}$
$y^{<2>}$
...
$y^{<T_y+1>}$
Transformer encoder layer
Transformer encoder-decoder layer
Transformer encoder layer
Transformer encoder-decoder layer
$x^{<1>}$
$x^{<T_x>}$
$y^{<1>}$
$y^{<T_y>}$

The left side is the encoder. The encoder attention layers do not have the causal mask.

The right side is the decoder. The decoder attention layers have the causal mask.

# ENCODER-DECODER



“素早い茶色のキツネが怠惰な犬を飛び越えます。”

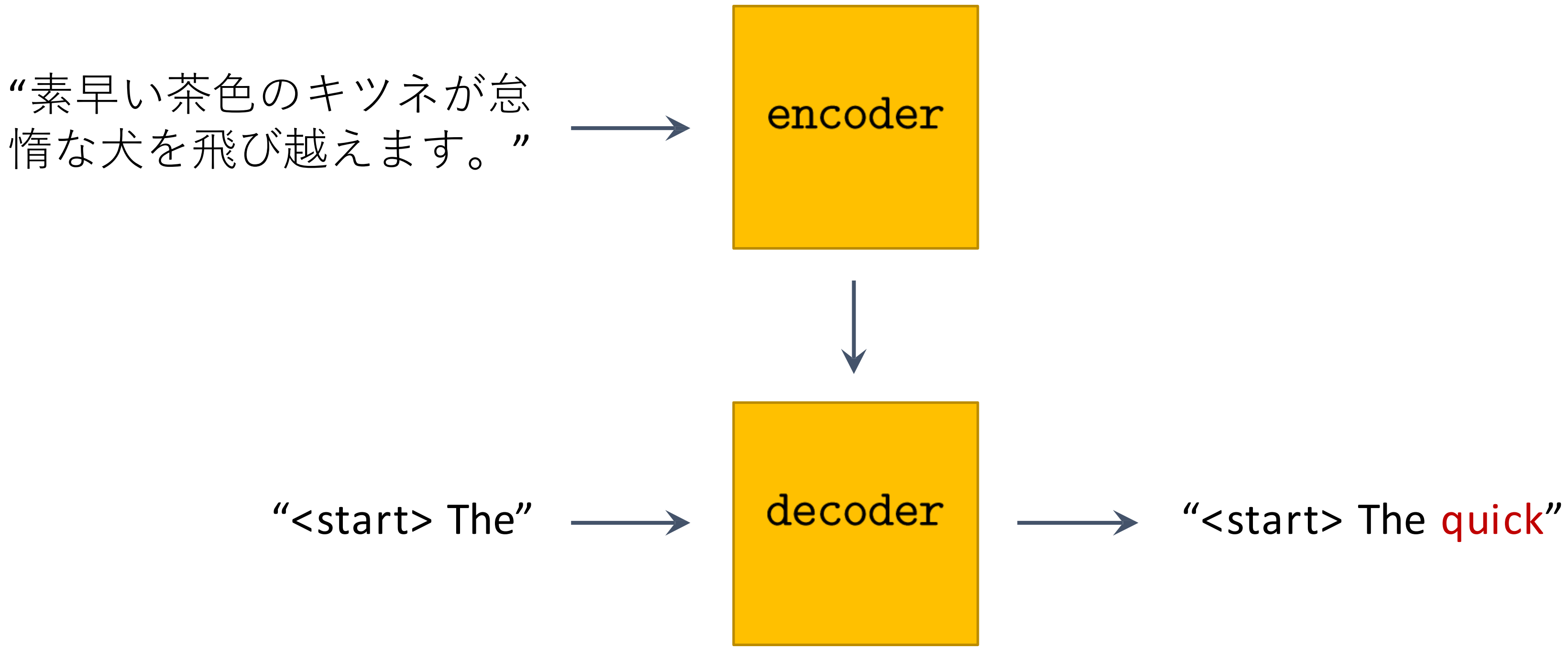
encoder

“<start>”

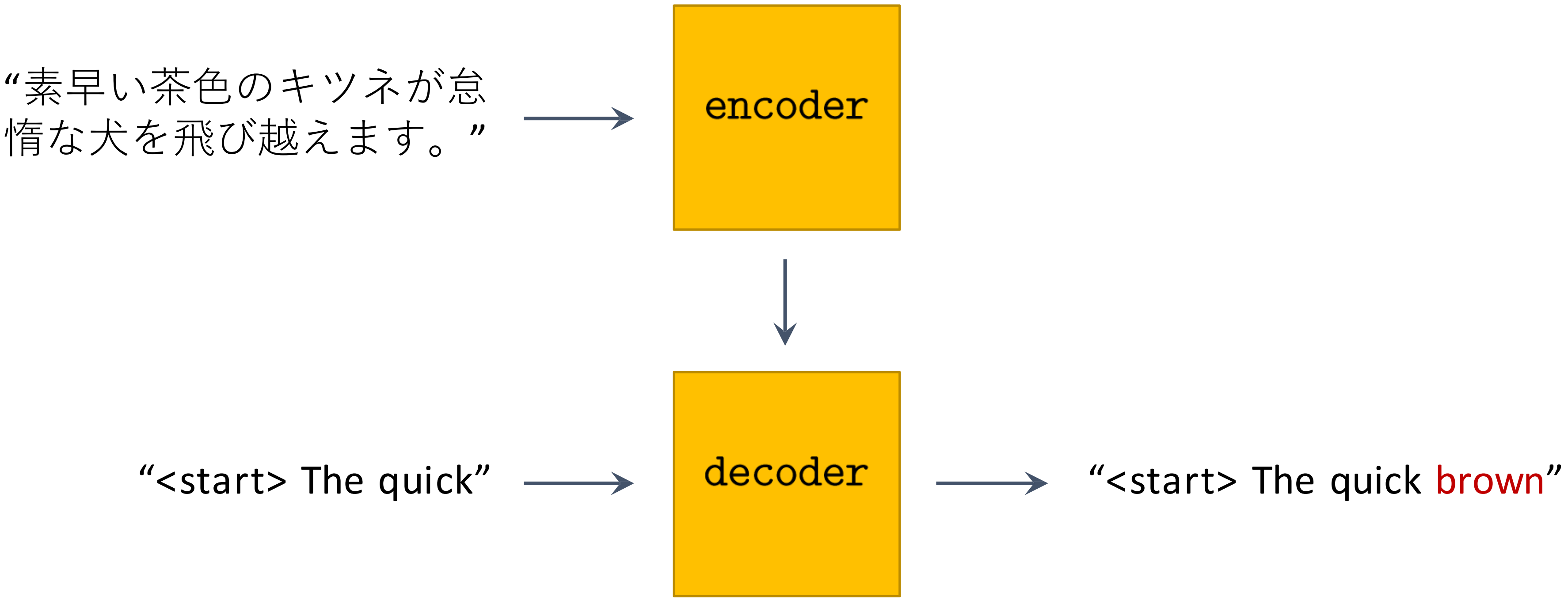
decoder

“<start> The”

# ENCODER-DECODER



# ENCODER-DECODER



“素早い茶色のキツネが怠惰な犬を飛び越えます。”

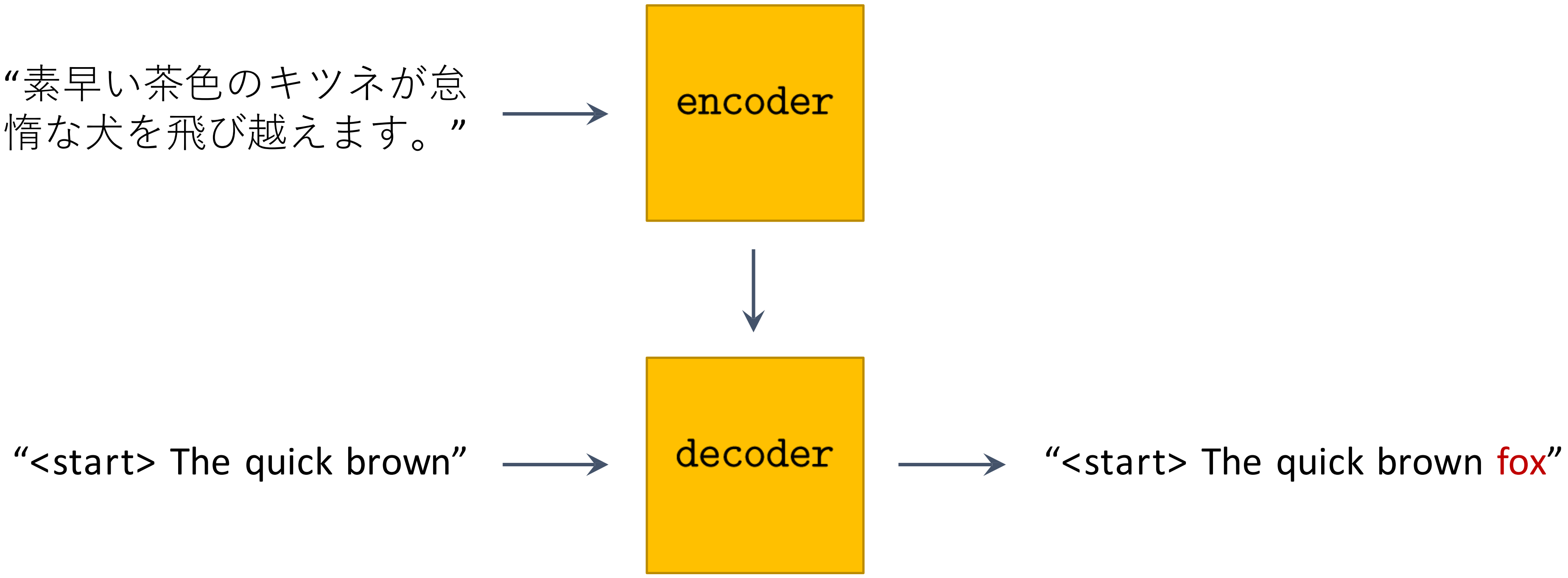
encoder

“<start> The quick”

decoder

“<start> The quick brown”

# ENCODER-DECODER



“素早い茶色のキツネが怠惰な犬を飛び越えます。”

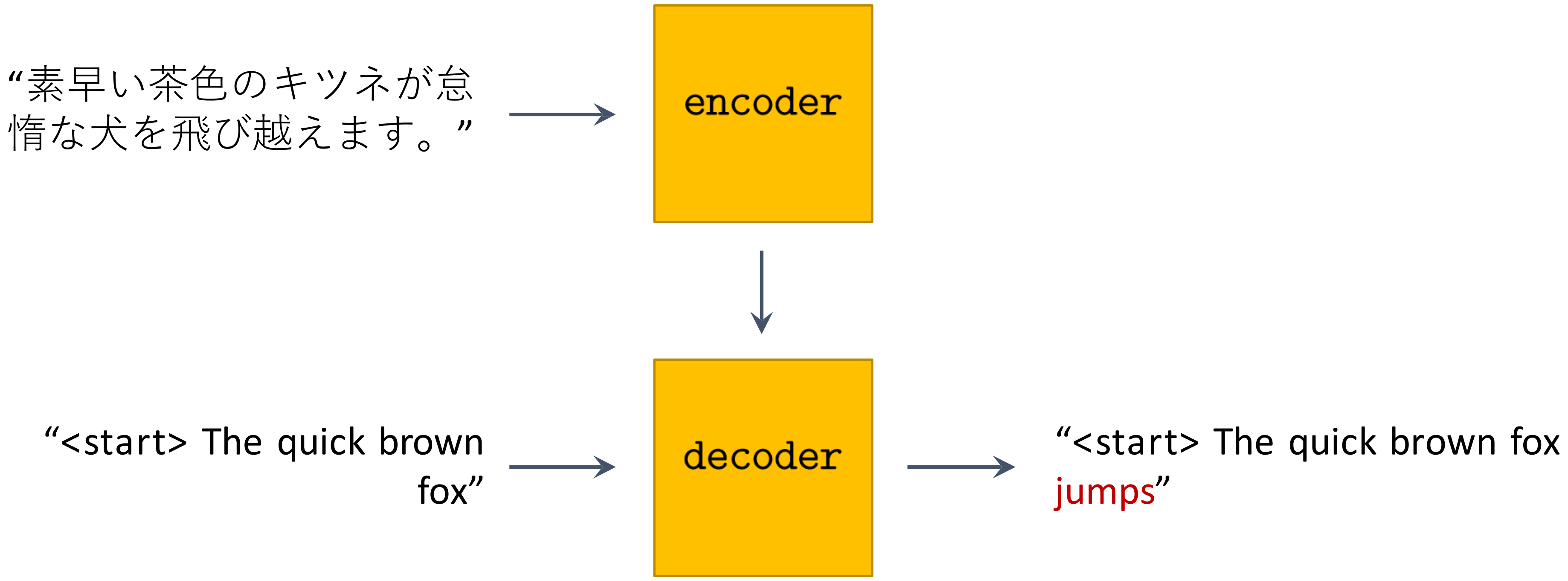
encoder

decoder

“<start> The quick brown”

“<start> The quick brown fox”

# ENCODER-DECODER



“素早い茶色のキツネが怠惰な犬を飛び越えます。”

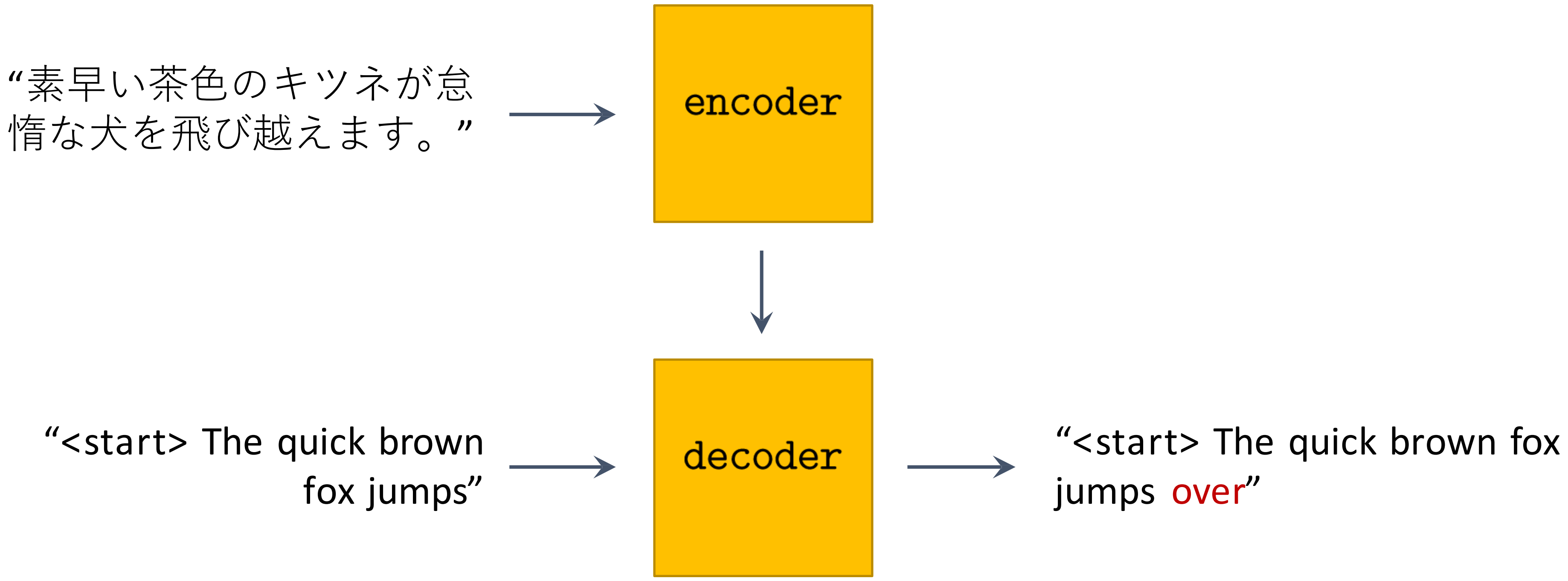
encoder

“<start> The quick brown fox”

decoder

“<start> The quick brown fox jumps”

# ENCODER-DECODER

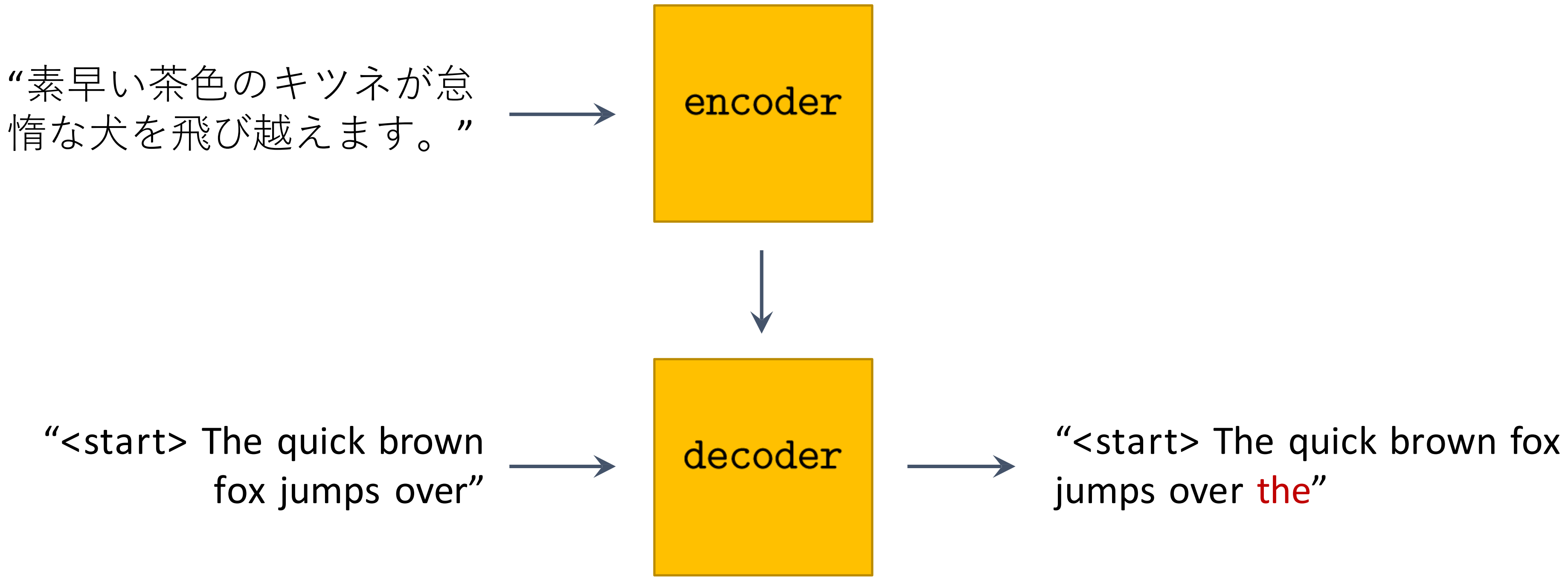


“素早い茶色のキツネが怠惰な犬を飛び越えます。”

“<start> The quick brown fox jumps”

“<start> The quick brown fox jumps over”

# ENCODER-DECODER



“素早い茶色のキツネが怠惰な犬を飛び越えます。”

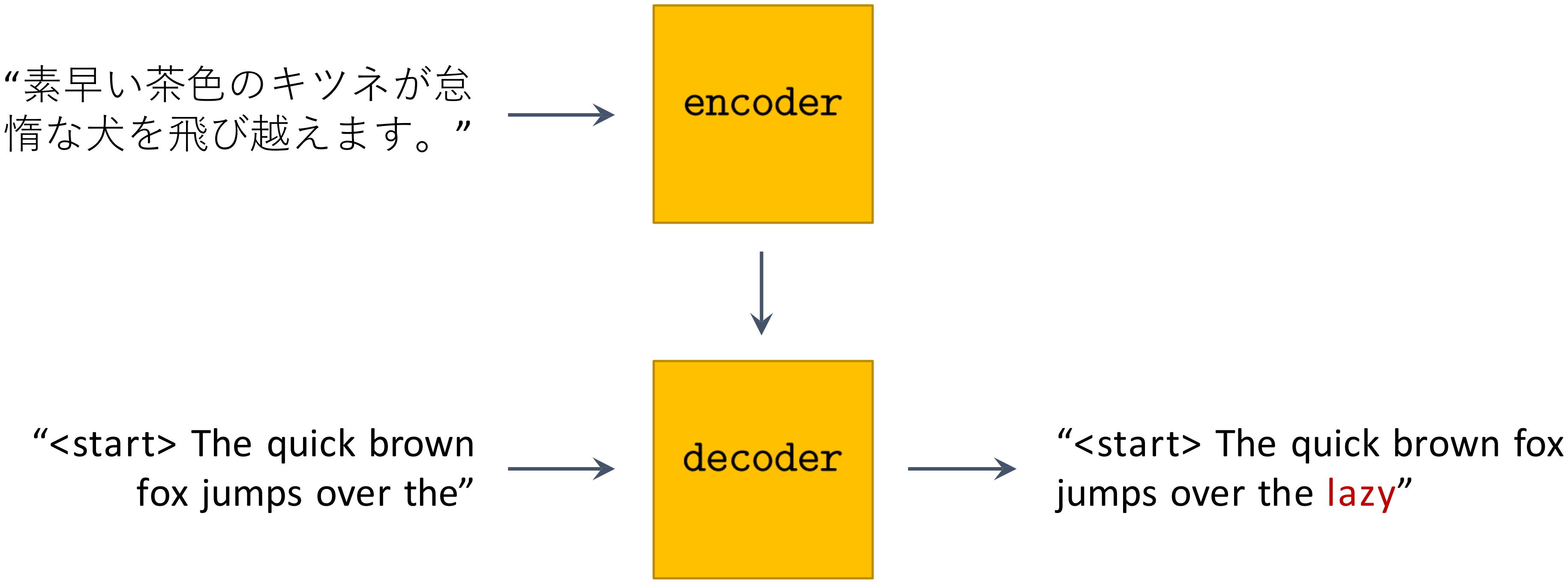
encoder

“<start> The quick brown fox jumps over”

decoder

“<start> The quick brown fox jumps over the”

# ENCODER-DECODER



“素早い茶色のキツネが怠惰な犬を飛び越えます。”

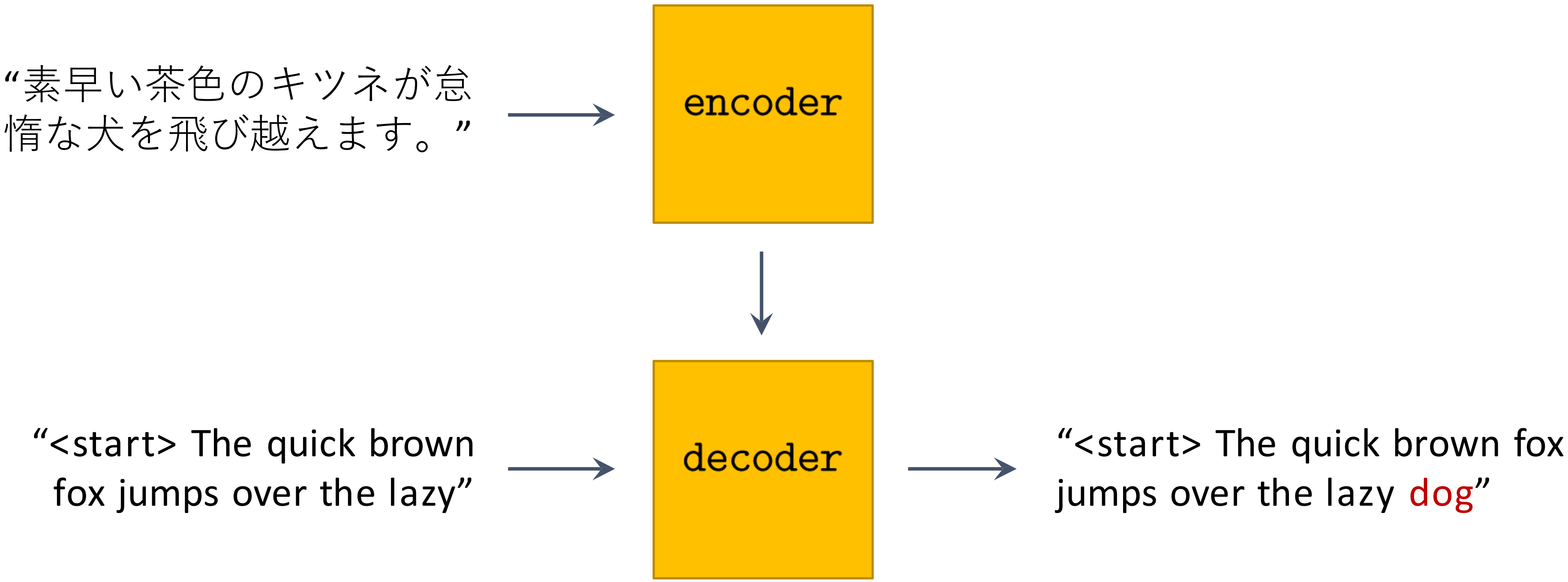
encoder

decoder

“<start> The quick brown fox jumps over the”

“<start> The quick brown fox jumps over the lazy”

# ENCODER-DECODER



“素早い茶色のキツネが怠惰な犬を飛び越えます。”

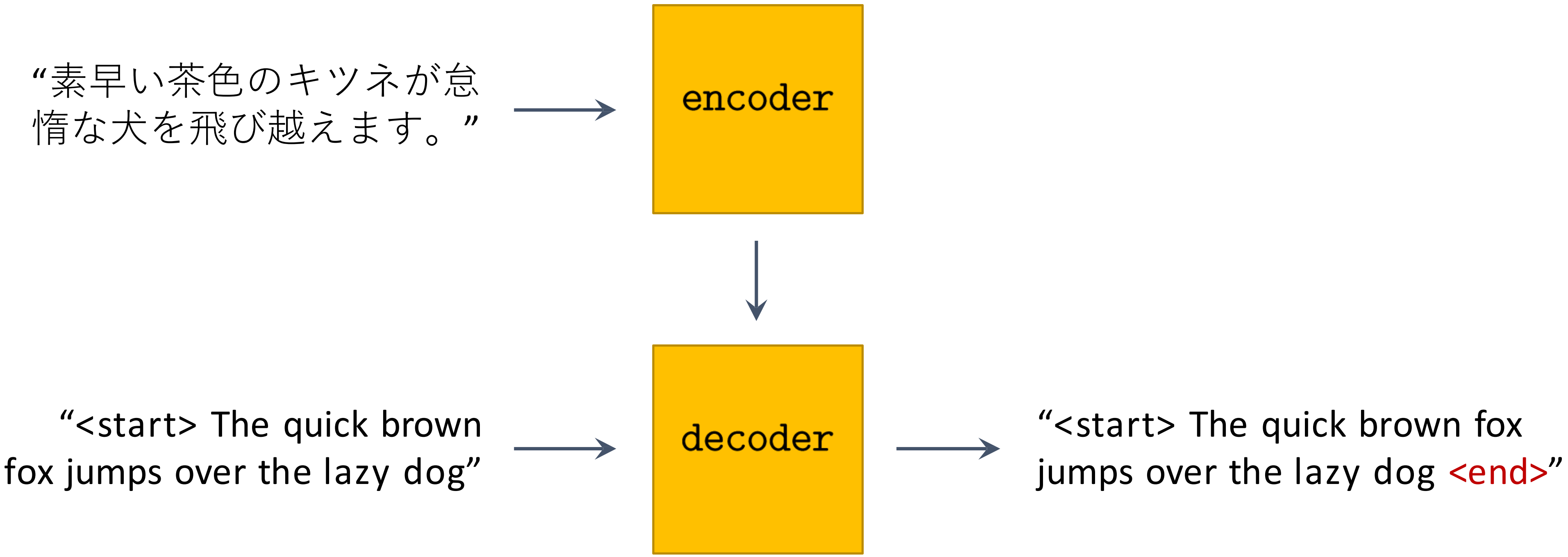
encoder

decoder

“<start> The quick brown fox jumps over the lazy”

“<start> The quick brown fox jumps over the lazy dog”

# ENCODER-DECODER



“素早い茶色のキツネが怠惰な犬を飛び越えます。”

encoder

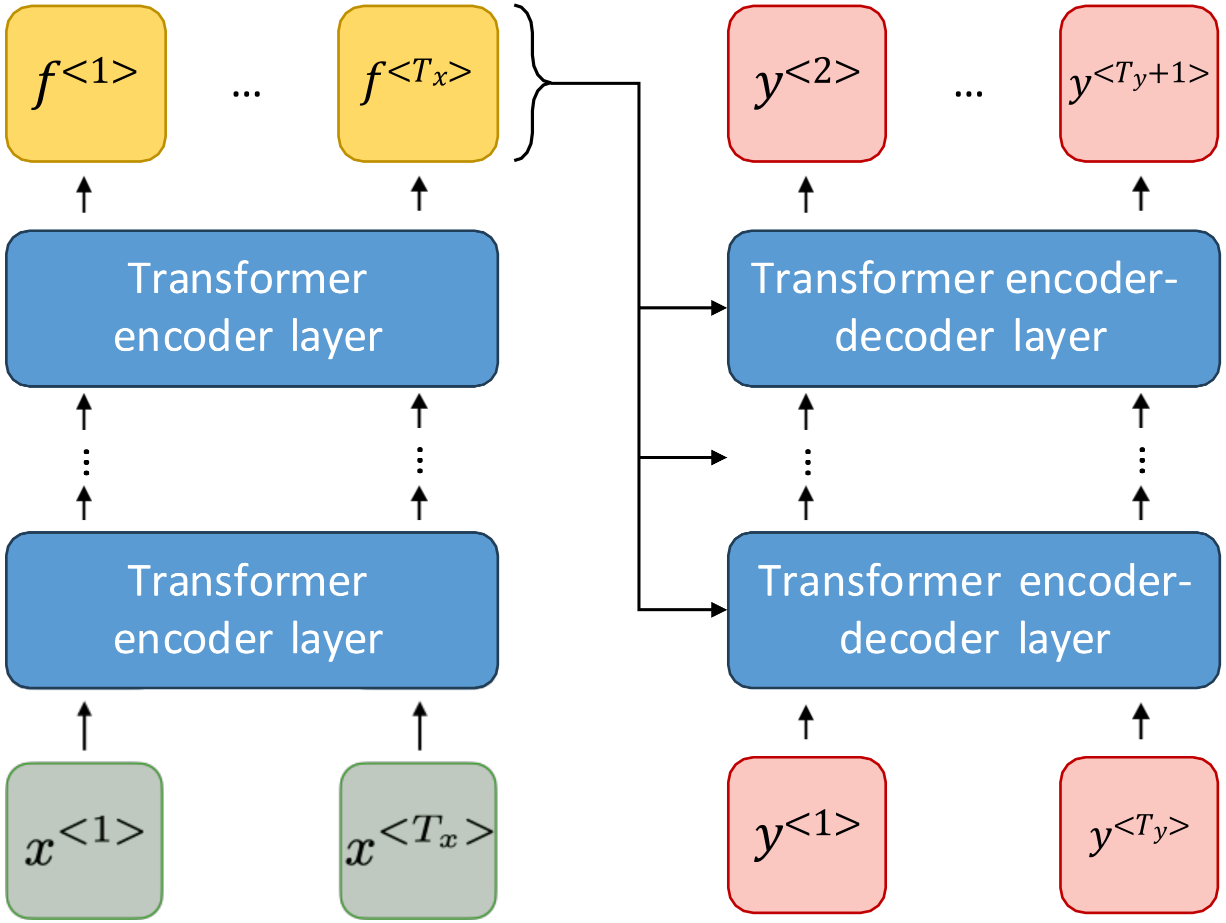
“<start> The quick brown fox jumps over the lazy dog”

decoder

“<start> The quick brown fox jumps over the lazy dog <end>”

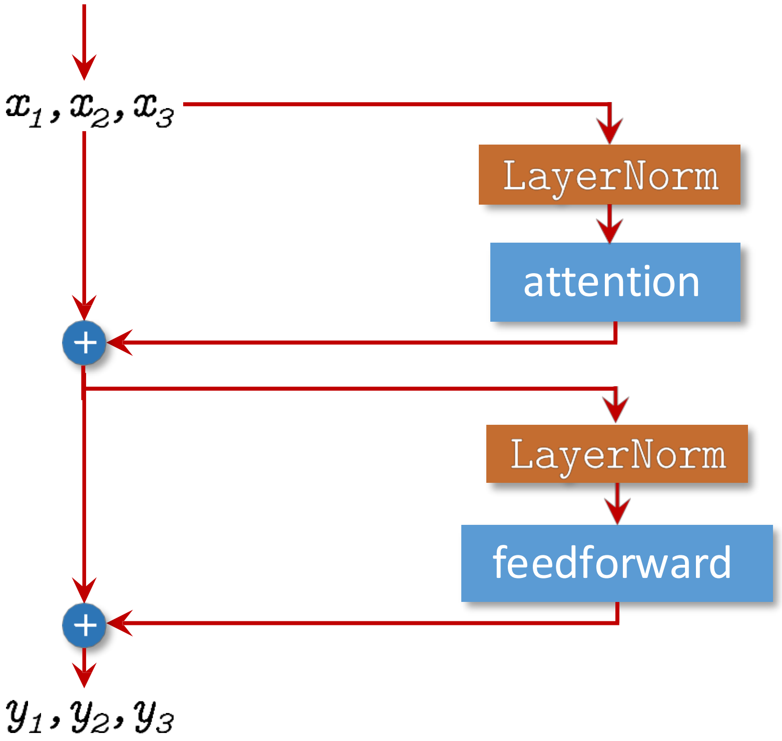
# TRANSFORMER ENCODER-DECODER

How does the transformer encoder-decoder layer incorporate information from the encoder (i.e., features)?



$f^{<1>}$
...
$f^{<T_x>}$
$y^{<2>}$
...
$y^{<T_y+1>}$
Transformer encoder layer
Transformer encoder-decoder layer
Transformer encoder layer
Transformer encoder-decoder layer
$x^{<1>}$
$x^{<T_x>}$
$y^{<1>}$
$y^{<T_y>}$

# TRANSFORMER LAYER

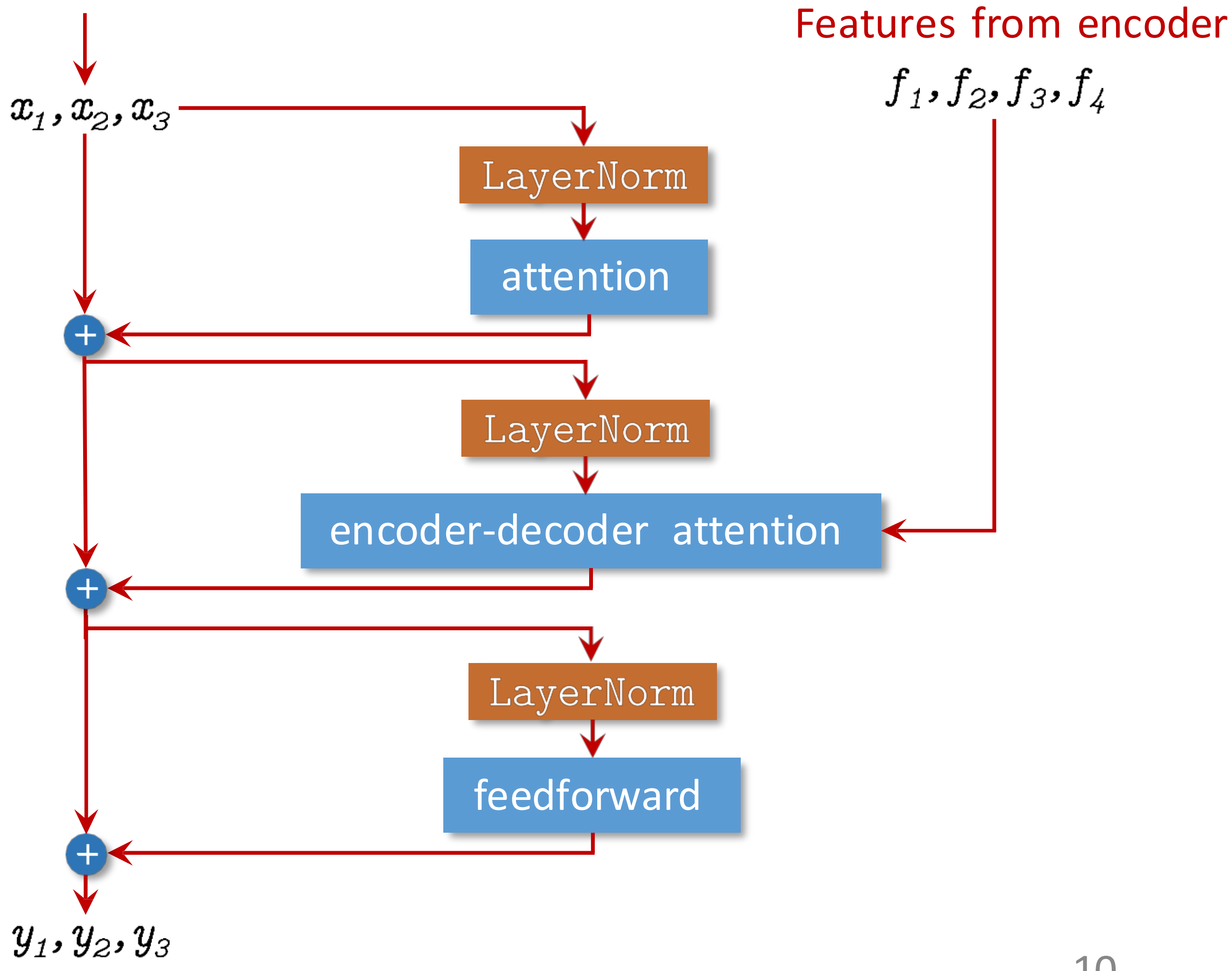


# TRANSFORMER ENCODER-DECODER LAYER

The first attention layer has a causal mask.

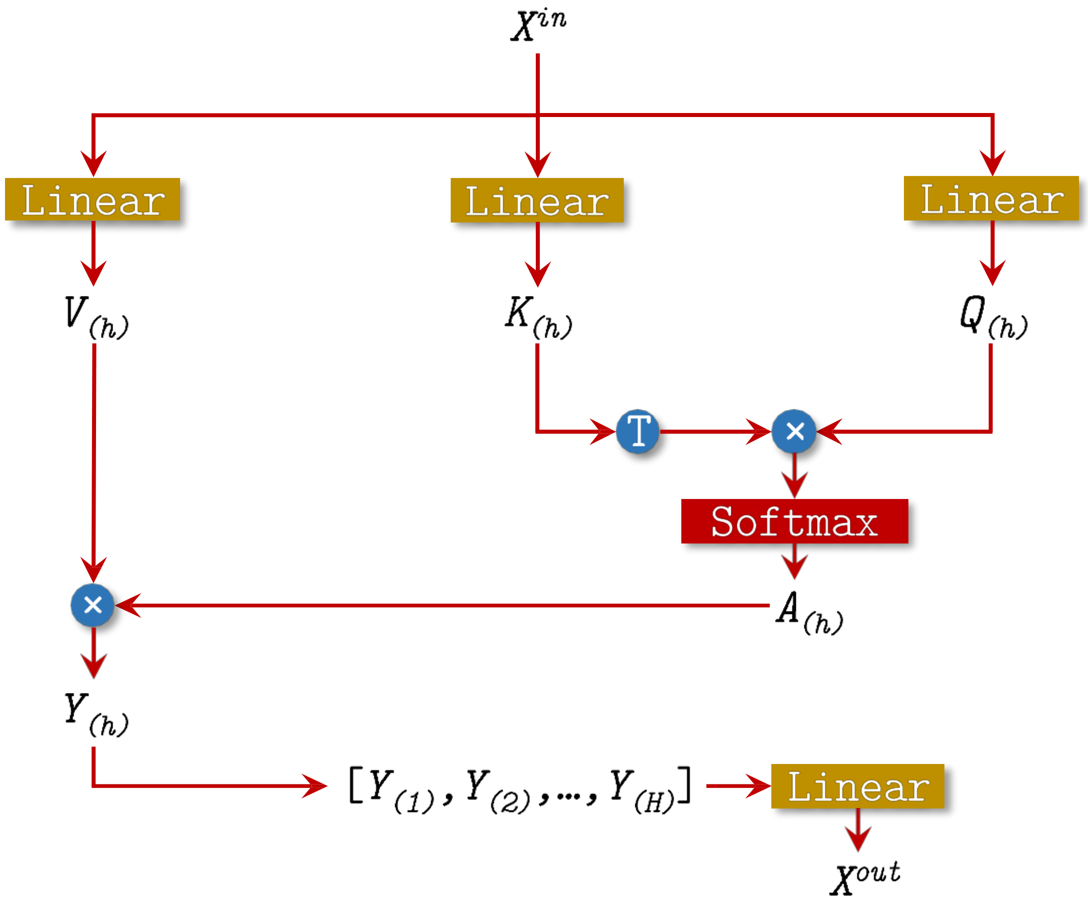
The encoder-decoder attention layer does not.

What does the encoder-decoder attention layer look like?



# ATTENTION LAYER

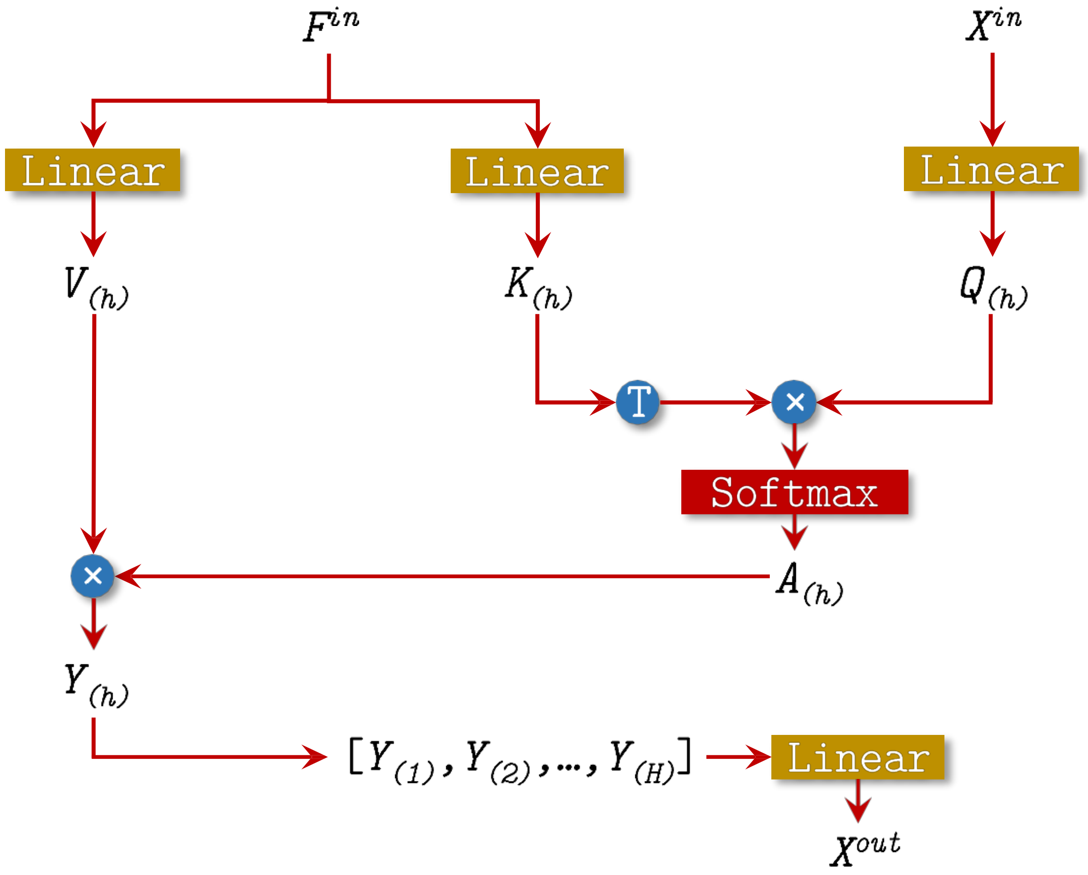
Recall this is the circuit diagram for the multi-head attention component.



# ENCODER-DECODER ATTENTION LAYER

$X^{in}$ are the inputs from the previous decoder layer.

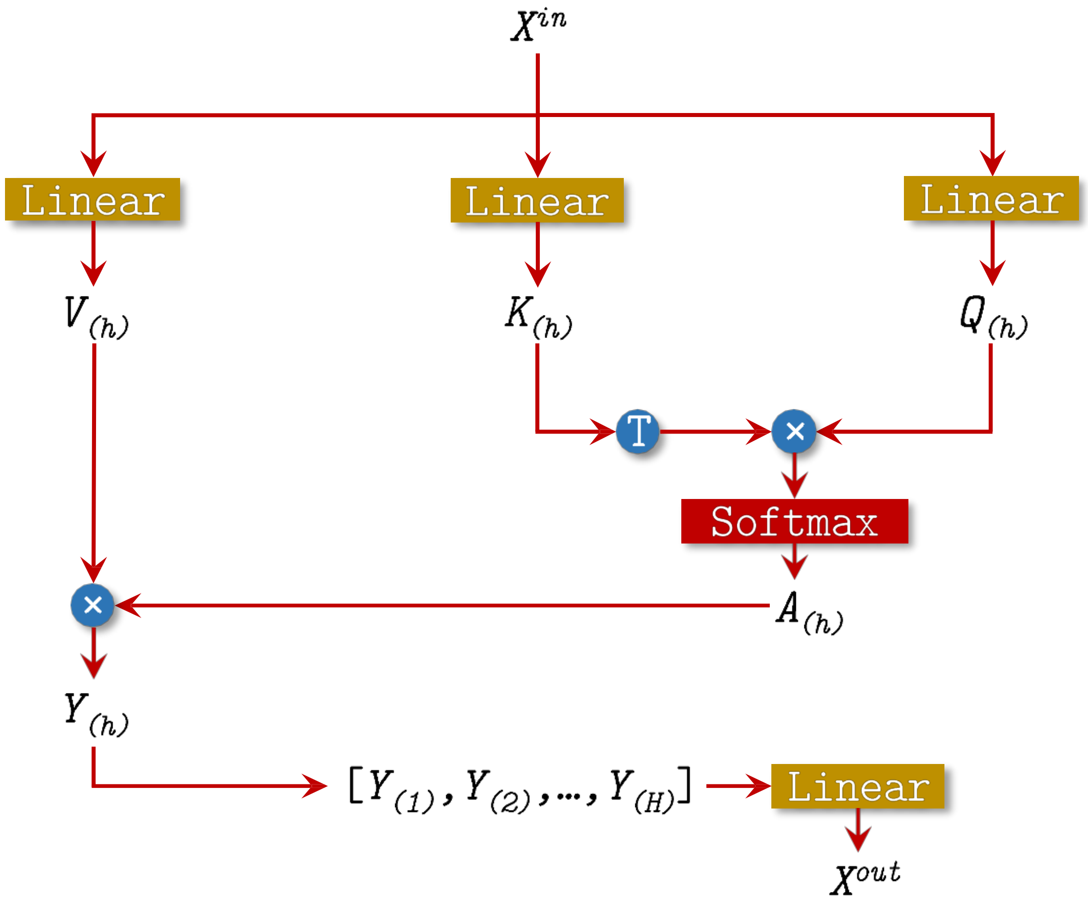
$F^{in}$ are the inputs from the encoder (i.e., features).



# ENCODER-ONLY AND DECODER-ONLY ATTENTION LAYER

So the attention layer where there is only 1 input is referred to as encoder-only or decoder-only.

Encoder-only attention layers do not have a causal mask.
Decoder-only attention layers have a causal mask.

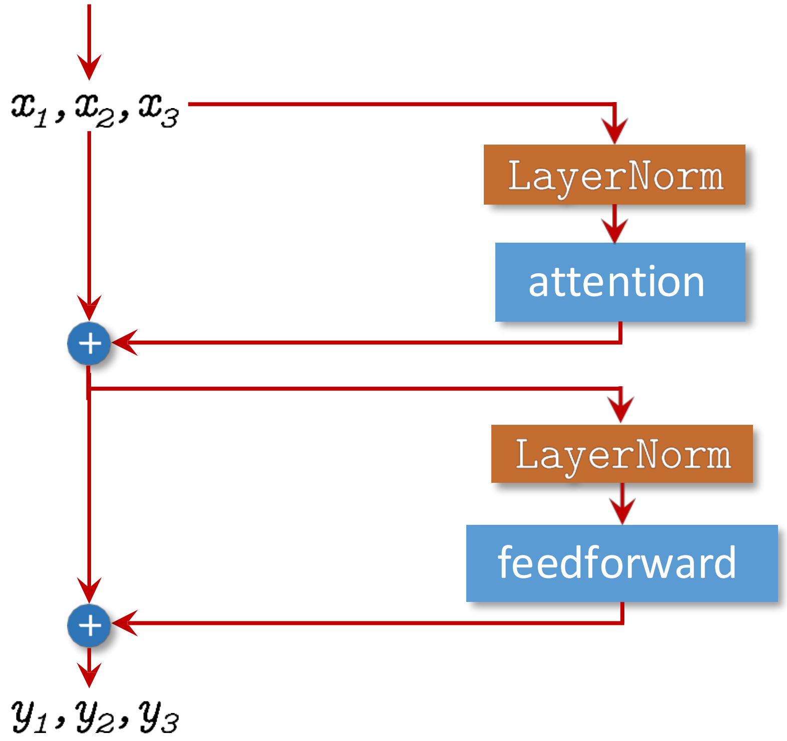


# ENCODER-ONLY AND DECODER-ONLY TRANSFORMER LAYER

Similarly, a transformer layer without encoder-decoder attention is called an encoder-only or decoder-only transformer layer,

depending on whether it has a causal mask.

Encoder-only transformers are also called bidirectional transformers.



# EXAMPLE ENCODER-DECODER TRANSFORMER MODELS

- Some example encoder-decoder models: (trivia)
  - BART (Lewis et al., 2019)
    - Up to 400M parameters.
  - T5 (Raffel et al., 2020)
    - Up to 11B parameters.
  - UnifiedQA (Khashabi et al., 2020)

# EXAMPLE ENCODER-ONLY TRANSFORMER MODELS

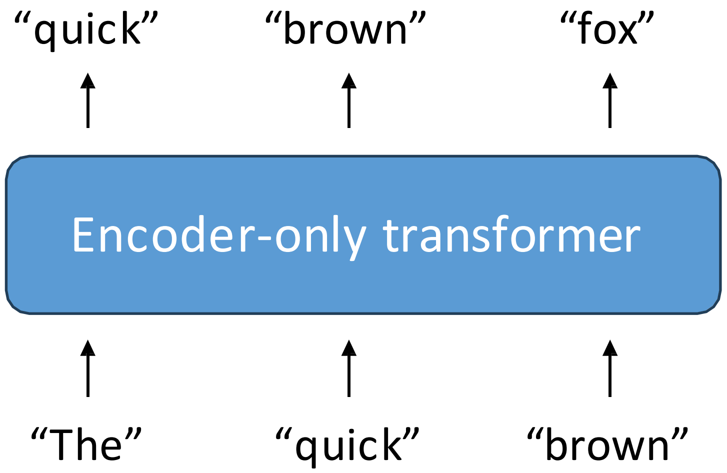
- Some example encoder-only models: (trivia)
  - BERT (Devlin et al., 2018)
    - Up to 355M parameters.
  - RoBERTa (Liu et al., 2019)
    - Up to 355M parameters.
  - DeBERTa (He et al., 2021)
    - Up to 1.5B parameters.

# EXAMPLE DECODER-ONLY TRANSFORMER MODELS

- Modern language models are decoder-only models. (trivia)
    - GPT-2 (Radford et al., 2019)
        - Up to 1.5B parameters.
    - GPT-3 (Brown et al., 2020)
        - Up to 175B parameters.
    - GPT-4 (OpenAI, 2023)
        - (unofficial) Up to 1.7T parameters (111B per expert).
    - LLaMA 3 (Meta, 2024)
        - Up to 405B parameters.
    - DeepSeek-V3 (DeepSeek-AI, 2024)
        - Up to 671B parameters (37B per expert).

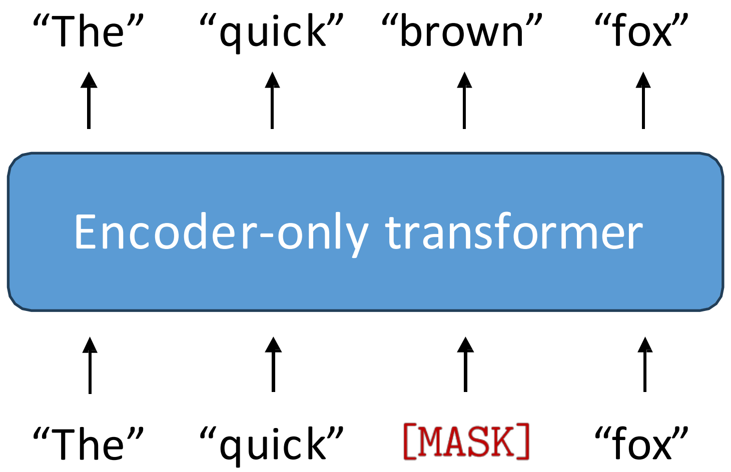
# TRAINING ENCODER-ONLY MODELS

- How are encoder-only models like BERT trained?
- We can’t use autoregressive language modeling since without the causal mask, encoder-only transformers can “cheat” by copying the next token.



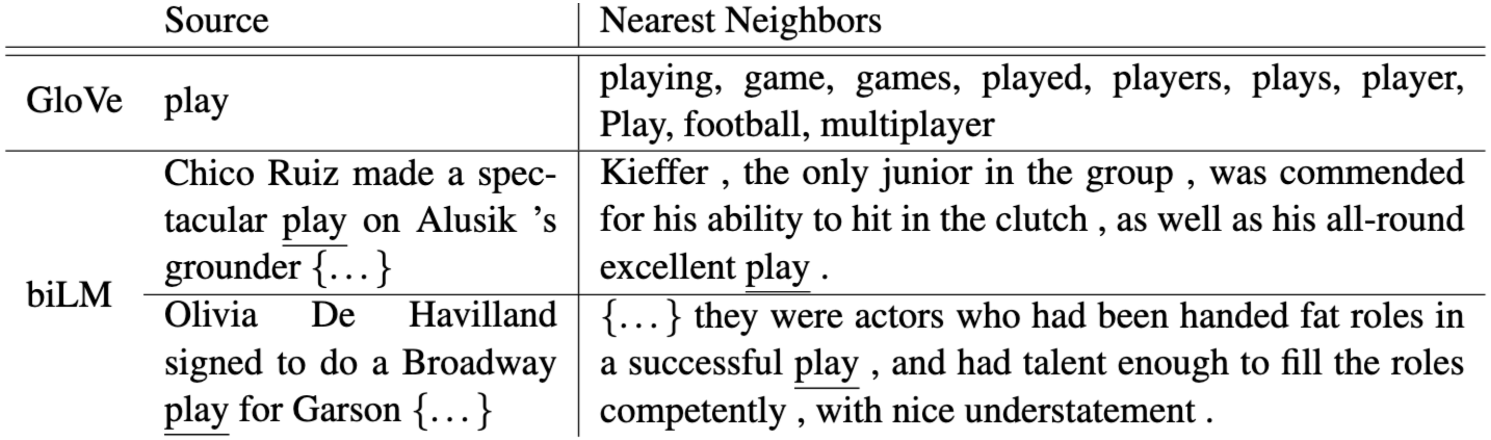
# TRAINING ENCODER-ONLY MODELS

- How are encoder-only models like BERT trained?
- Measure the loss function only on the masked tokens.



# Transformers vs. LSTMs

- RNN based architectures hard code sequential assumptions into the network structure.
- ELMo was an attempt for large scale pre-training of an LSTM based **contextualized word embedding** model.

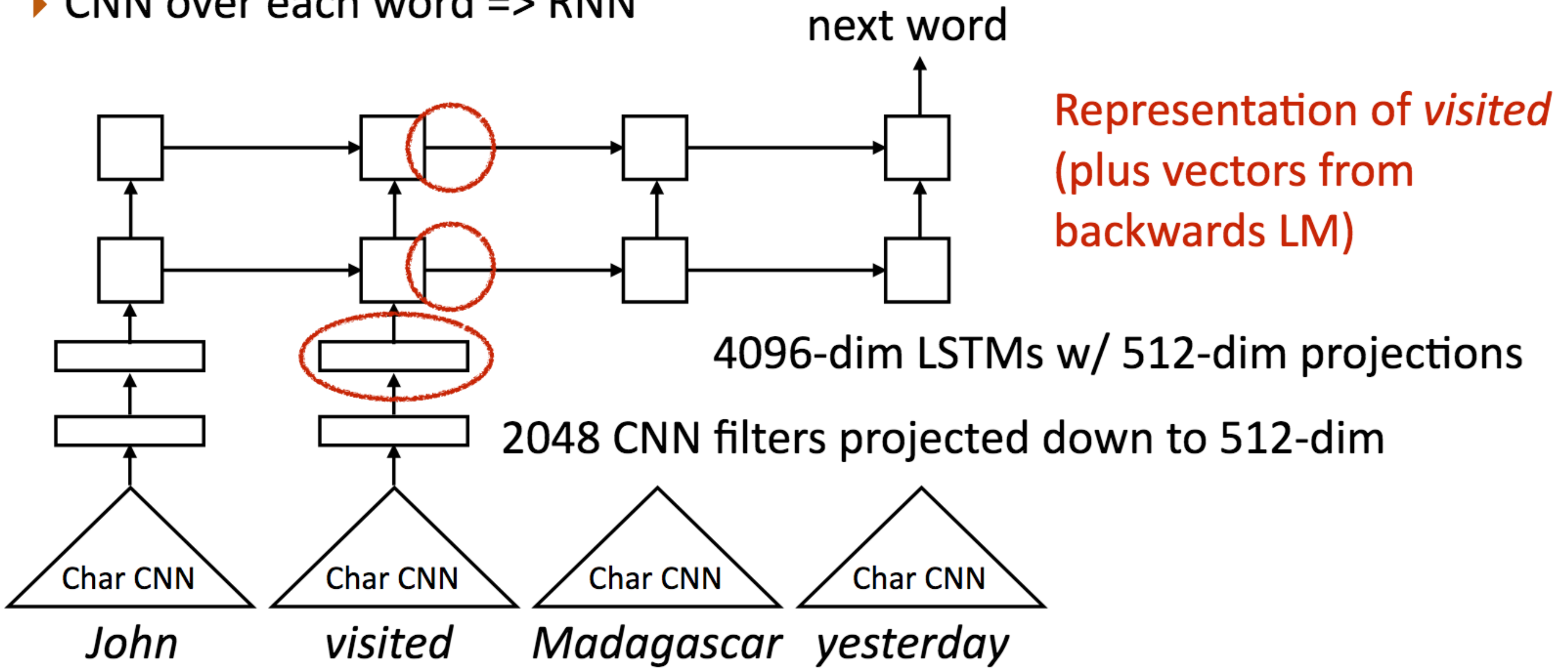


|  | Source | Nearest Neighbors |
|---|---|---|
| GloVe | play | playing, game, games, played, players, plays, player, Play, football, multiplayer |
| biLM | Chico Ruiz made a spectacular play on Alusik 's grounder {. . . } | Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play . |
|  | Olivia De Havilland signed to do a Broadway play for Garson {. . . } | {. . . } they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement . |

Deep contextualized word representations. Peters et-al 2018

# ELMo Architecture

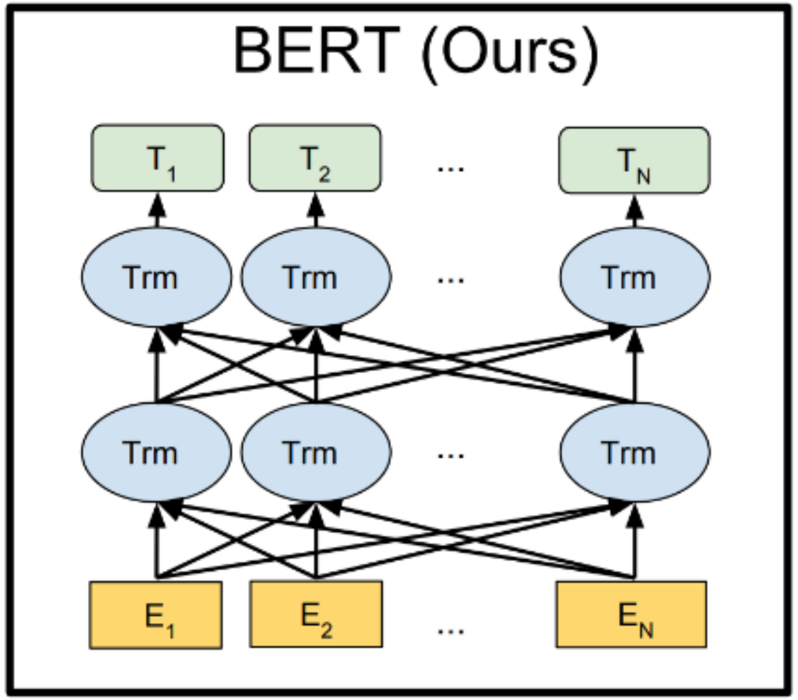
- CNN over each word => RNN


next word
Representation of *visited* (plus vectors from backwards LM)
4096-dim LSTMs w/ 512-dim projections
2048 CNN filters projected down to 512-dim
Char CNN
Char CNN
Char CNN
Char CNN
*John*
*visited*
*Madagascar*
*yesterday*

Peters et al. (2018)

# BERT

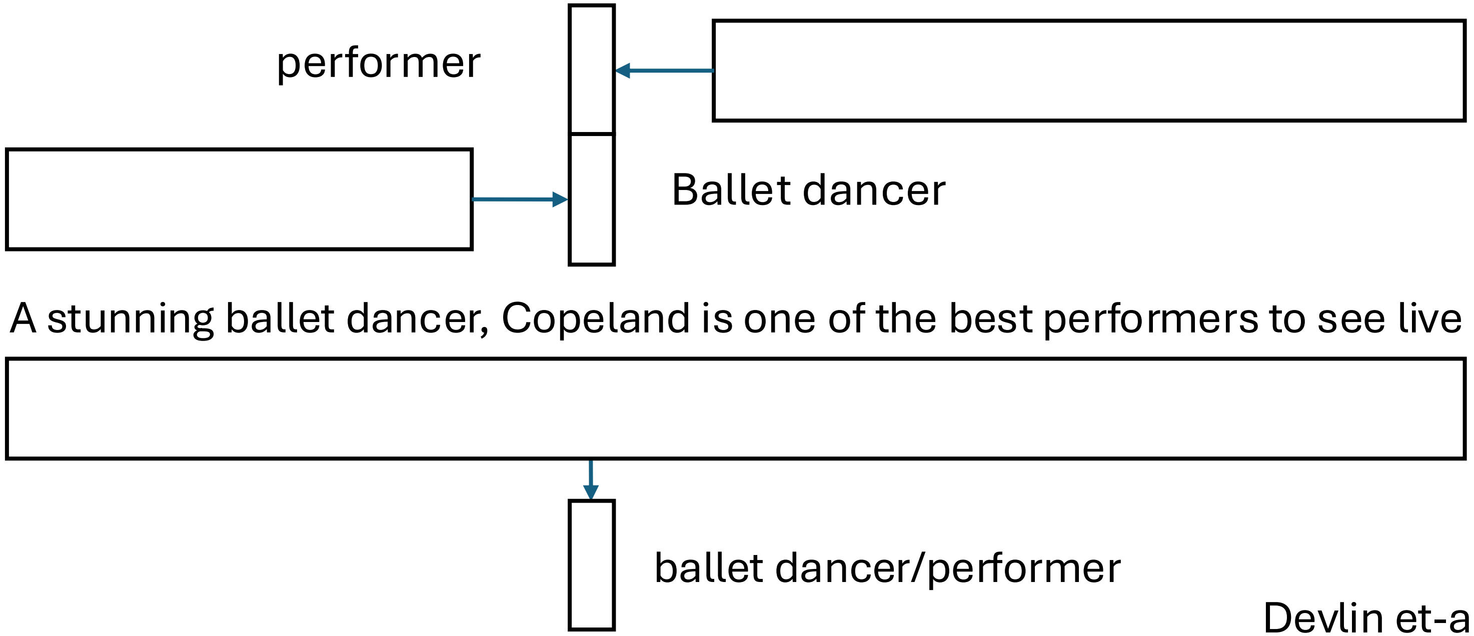
- Transformer-based approach instead of an LSTM -based like ELMo.
  - Transformer vs. LSTM
  - Masked language objective instead of usual LM
  - Fine-tuned at test time



BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding Devlin et-al 2019

# BERT

Note the difference: traditional LM uses ”one-sided” information, BERT uses attention from the entire sentence (interactions between ”past and future” tokens is modeled.


performer
Ballet dancer
A stunning ballet dancer, Copeland is one of the best performers to see live
ballet dancer/performer

Devlin et-al 2019

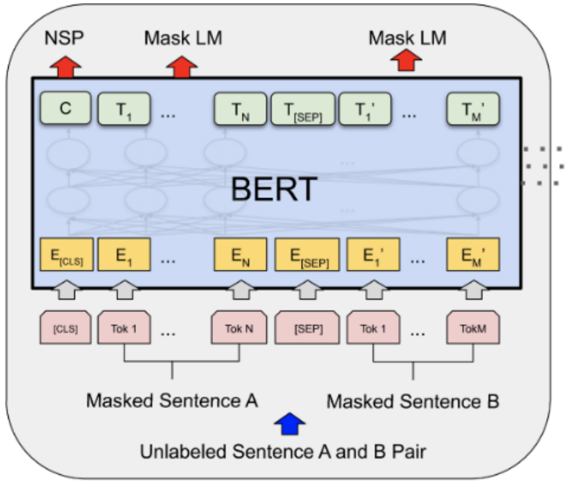
# Architecture

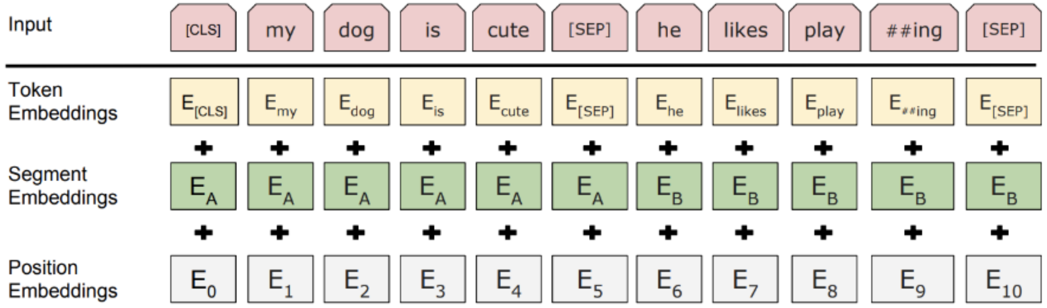
BERT Base: 12 layers, 768-dim per wordpiece token, 12 heads. Total params = 110M

BERT Large: 24 layers, 1024-dim per wordpiece token, 16 heads. Total params = 340M

Positional embeddings and segment embeddings, 30k word pieces

This is the model that gets **pre-trained** on a large corpus





| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Input | [CLS] | my | dog | is | cute | [SEP] | he | likes | play | ##ing | [SEP] |
| Token Embeddings | $E_{[CLS]}$ | $E_{my}$ | $E_{dog}$ | $E_{is}$ | $E_{cute}$ | $E_{[SEP]}$ | $E_{he}$ | $E_{likes}$ | $E_{play}$ | $E_{\#\#ing}$ | $E_{[SEP]}$ |
| | + | + | + | + | + | + | + | + | + | + | + |
| Segment Embeddings | $E_A$ | $E_A$ | $E_A$ | $E_A$ | $E_A$ | $E_A$ | $E_B$ | $E_B$ | $E_B$ | $E_B$ | $E_B$ |
| | + | + | + | + | + | + | + | + | + | + | + |
| Position Embeddings | $E_0$ | $E_1$ | $E_2$ | $E_3$ | $E_4$ | $E_5$ | $E_6$ | $E_7$ | $E_8$ | $E_9$ | $E_{10}$ |

Devlin et al. (2019)

# Comparing Contextualized Representations

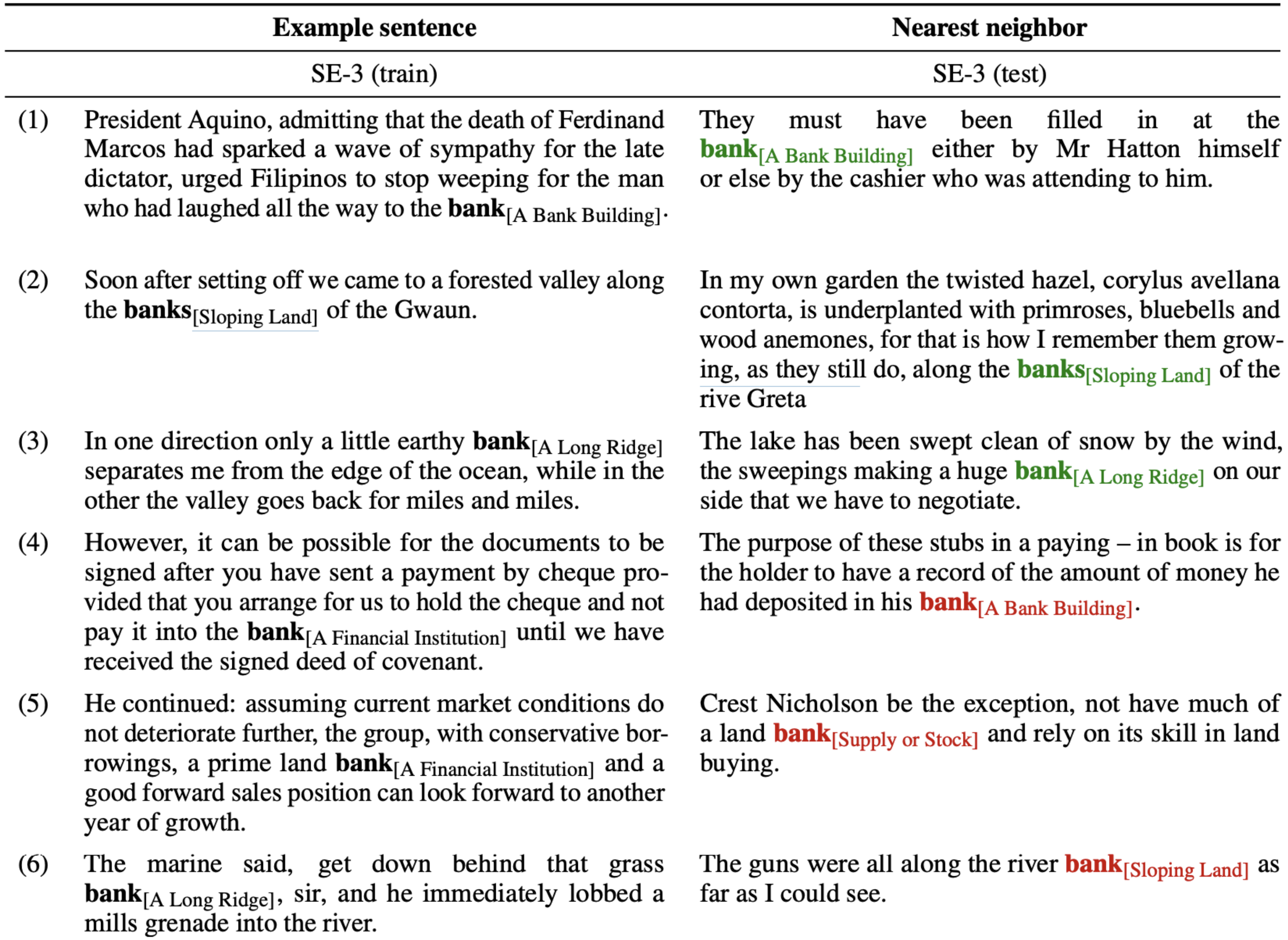
- Consider the word **bank**, how many different meaning does it have?
  - ”the river **bank**”, “the **bank** refused a loan”, “the **bank** on the corner of main st”, etc.
- Words can have different senses, **word sense disambiguation** is the task of mapping a word to the right sense.
- Heavily depends on the context in which the word appears.

- **Question**: how well would different context representations do on this task?

Wiedemann et-al 2019

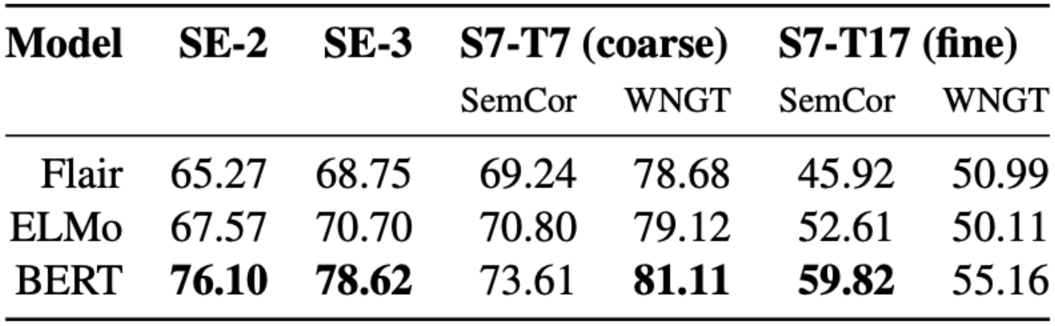
# Benchmarking WSD

- Prediction is done using KNN, given the training data.
- Use different embeddings to implement a similarity function.
- ”good” representations should identify the similarity in context of words with similar senses.



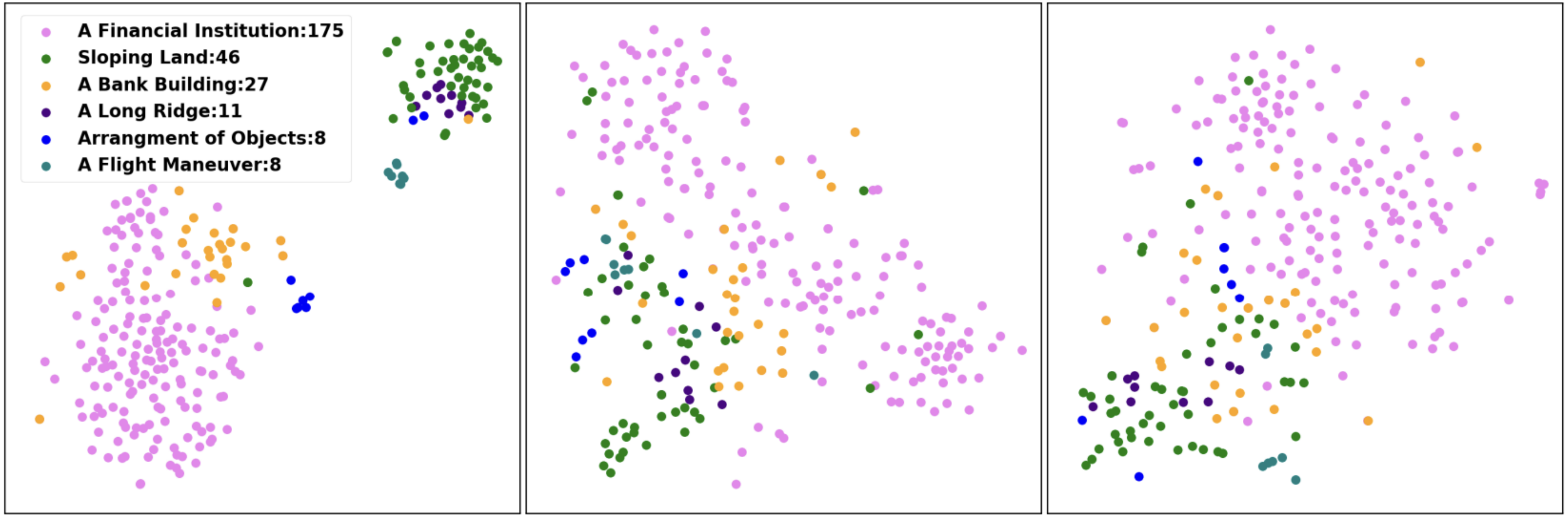
| | Example sentence | Nearest neighbor |
|---|---|---|
| | SE-3 (train) | SE-3 (test) |
| (1) | President Aquino, admitting that the death of Ferdinand Marcos had sparked a wave of sympathy for the late dictator, urged Filipinos to stop weeping for the man who had laughed all the way to the **bank**$_{\text{[A Bank Building]}}$. | They must have been filled in at the **bank**$_{\text{[A Bank Building]}}$ either by Mr Hatton himself or else by the cashier who was attending to him. |
| (2) | Soon after setting off we came to a forested valley along the **banks**$_{\text{[Sloping Land]}}$ of the Gwaun. | In my own garden the twisted hazel, corylus avellana contorta, is underplanted with primroses, bluebells and wood anemones, for that is how I remember them growing, as they still do, along the **banks**$_{\text{[Sloping Land]}}$ of the rive Greta |
| (3) | In one direction only a little earthy **bank**$_{\text{[A Long Ridge]}}$ separates me from the edge of the ocean, while in the other the valley goes back for miles and miles. | The lake has been swept clean of snow by the wind, the sweepings making a huge **bank**$_{\text{[A Long Ridge]}}$ on our side that we have to negotiate. |
| (4) | However, it can be possible for the documents to be signed after you have sent a payment by cheque provided that you arrange for us to hold the cheque and not pay it into the **bank**$_{\text{[A Financial Institution]}}$ until we have received the signed deed of covenant. | The purpose of these stubs in a paying – in book is for the holder to have a record of the amount of money he had deposited in his **bank**$_{\text{[A Bank Building]}}$. |
| (5) | He continued: assuming current market conditions do not deteriorate further, the group, with conservative borrowings, a prime land **bank**$_{\text{[A Financial Institution]}}$ and a good forward sales position can look forward to another year of growth. | Crest Nicholson be the exception, not have much of a land **bank**$_{\text{[Supply or Stock]}}$ and rely on its skill in land buying. |
| (6) | The marine said, get down behind that grass **bank**$_{\text{[A Long Ridge]}}$, sir, and he immediately lobbed a mills grenade into the river. | The guns were all along the river **bank**$_{\text{[Sloping Land]}}$ as far as I could see. |

# Benchmarking WSD



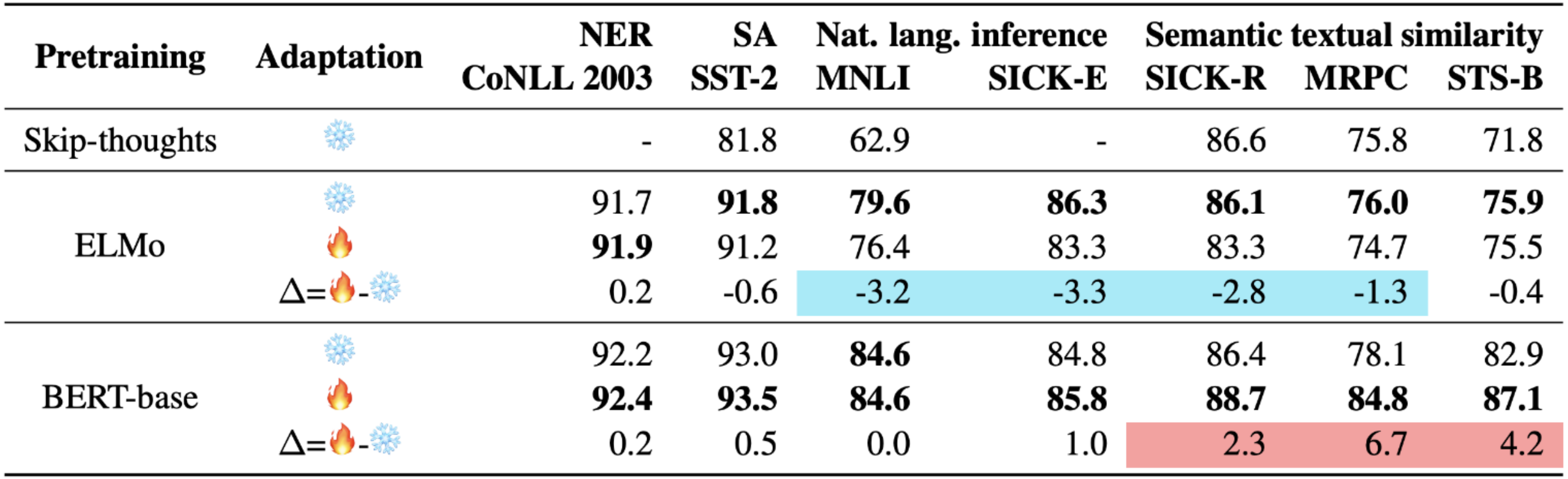
| Model | SE-2 | SE-3 | S7-T7 (coarse) | | S7-T17 (fine) | |
|---|---|---|---|---|---|---|
| | | | SemCor | WNGT | SemCor | WNGT |
| Flair | 65.27 | 68.75 | 69.24 | 78.68 | 45.92 | 50.99 |
| ELMo | 67.57 | 70.70 | 70.80 | 79.12 | 52.61 | 50.11 |
| BERT | **76.10** | **78.62** | 73.61 | **81.11** | **59.82** | 55.16 |

Table 2: kNN with $k = 1$ WSD performance (F1%). Best results for each testset are marked bold.



(a) BERT (b) Flair (c) ELMo

# Fine Tuning for downstream tasks



| Pretraining | Adaptation | NER | SA | Nat. lang. inference | | Semantic textual similarity | | |
|---|---|---|---|---|---|---|---|---|
| | | CoNLL 2003 | SST-2 | MNLI | SICK-E | SICK-R | MRPC | STS-B |
| Skip-thoughts | ❄️ | - | 81.8 | 62.9 | - | 86.6 | 75.8 | 71.8 |
| ELMo | ❄️ | 91.7 | **91.8** | **79.6** | **86.3** | **86.1** | **76.0** | **75.9** |
| | 🔥 | **91.9** | 91.2 | 76.4 | 83.3 | 83.3 | 74.7 | 75.5 |
| | Δ=🔥-❄️ | 0.2 | -0.6 | -3.2 | -3.3 | -2.8 | -1.3 | -0.4 |
| BERT-base | ❄️ | 92.2 | 93.0 | **84.6** | 84.8 | 86.4 | 78.1 | 82.9 |
| | 🔥 | **92.4** | **93.5** | **84.6** | **85.8** | **88.7** | **84.8** | **87.1** |
| | Δ=🔥-❄️ | 0.2 | 0.5 | 0.0 | 1.0 | 2.3 | 6.7 | 4.2 |

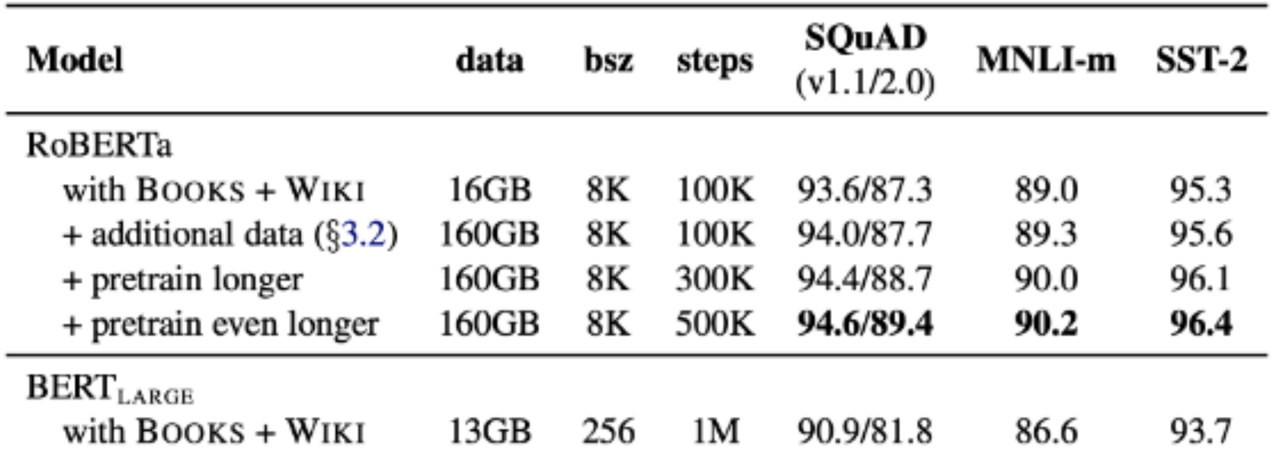
BERT tends to perform better if the entire network is trained, unlike ELMO

# RoBERTa

- "Robustly optimized BERT"

- 160GB of data instead of 16 GB

- Dynamic masking: standard BERT uses the same MASK scheme for every epoch, RoBERTa recomputes them

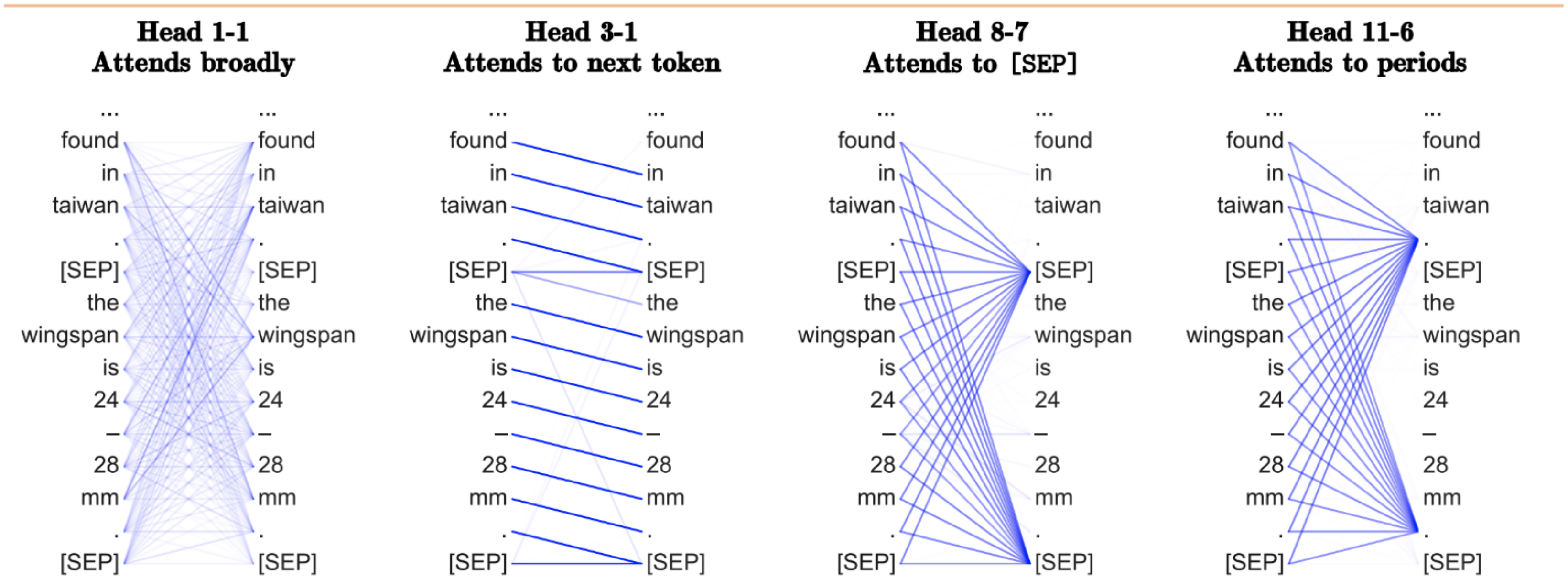


| Model | data | bsz | steps | SQuAD (v1.1/2.0) | MNLI-m | SST-2 |
|---|---|---|---|---|---|---|
| RoBERTa | | | | | | |
| with BOOKS + WIKI | 16GB | 8K | 100K | 93.6/87.3 | 89.0 | 95.3 |
| + additional data (§3.2) | 160GB | 8K | 100K | 94.0/87.7 | 89.3 | 95.6 |
| + pretrain longer | 160GB | 8K | 300K | 94.4/88.7 | 90.0 | 96.1 |
| + pretrain even longer | 160GB | 8K | 500K | **94.6/89.4** | **90.2** | **96.4** |
| $\text{BERT}_{\text{LARGE}}$ | | | | | | |
| with BOOKS + WIKI | 13GB | 256 | 1M | 90.9/81.8 | 86.6 | 93.7 |

- New training + more data = better performance

Liu et al. (2019)

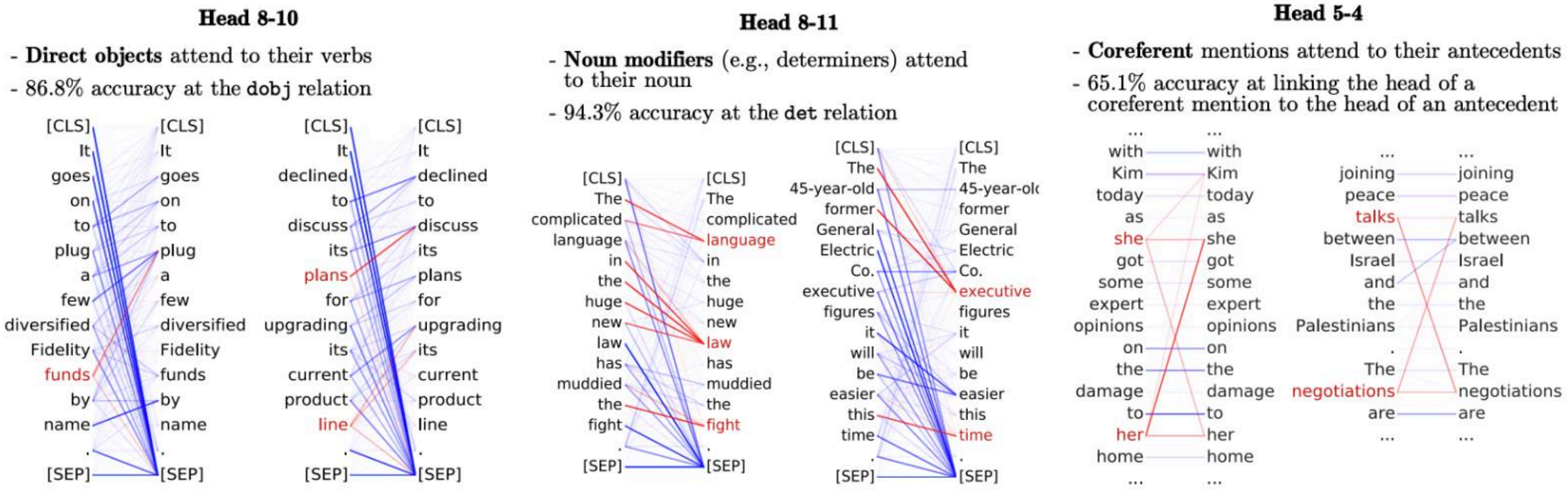
# What does BERT learn?



- Heads on transformers learn interesting and diverse things: content heads (attend based on content), positional heads (based on position), etc.

Clark et al. (2019)

# What does BERT learn?



**Head 8-10**

- **Direct objects** attend to their verbs
- 86.8% accuracy at the `dobj` relation

**Head 8-11**

- **Noun modifiers** (e.g., determiners) attend to their noun
- 94.3% accuracy at the `det` relation

**Head 5-4**

- **Coreferent** mentions attend to their antecedents
- 65.1% accuracy at linking the head of a coreferent mention to the head of an antecedent

[CLS] It goes on to plug a few diversified Fidelity funds by name . [SEP]
[CLS] It declined to discuss its plans for upgrading its current product line . [SEP]
[CLS] The complicated language in the huge new law has muddied the fight . [SEP]
[CLS] The 45-year-old former General Electric Co. executive figures it will be easier this time . [SEP]
... with Kim today as she got some expert opinions on the damage to her home ...
... joining peace talks between Israel and the Palestinians . The negotiations are ...

- Still way worse than what supervised systems can do, but interesting that this is learned organically

# QUESTIONS?