Attention & Transformers for NLP Tasks

By: Ben Paulson & John Cisler

Our Project

- **What We Looked At**
  - “Attention Is All You Need” (2017)
  - Language Translation (English to French)
- **Motivation**
  - Heavy usage in modern systems

Existing Use Cases

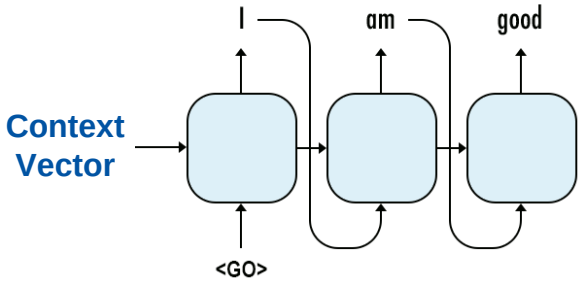
- Google Translate
- Google's BERT
  - Google Search
  - Facebook's RoBERTa
- GPT
  - Chat
  - Standford LLaMA (local GPT)

Background - NLP

- **Language Translation, Sentiment Analysis, Answering Questions, Text Summarization, etc…**
- **Token:** String of values representing a text unit
- **Encoding:** Deterministic mapping of token
- **Embedding space:** All vectorized tokens; closer means similar meaning
  - Accounts for semantics (e.g. ”right”)

Background – Sequence Models

- Strung together element level models
- Requires sequence context and input
  - Context vector for holding information



Background – Attention

- **Pay attention to portions of the input sequence (>1000 tokens)**
- **Types (for our purposes):**
  - **Self –** focus on relevant parts of input + capture dependencies
  - **Multi-headed –** extension of self; learn different aspects of input simultaneously
    - Learn nuance + complex dependencies

Transformer

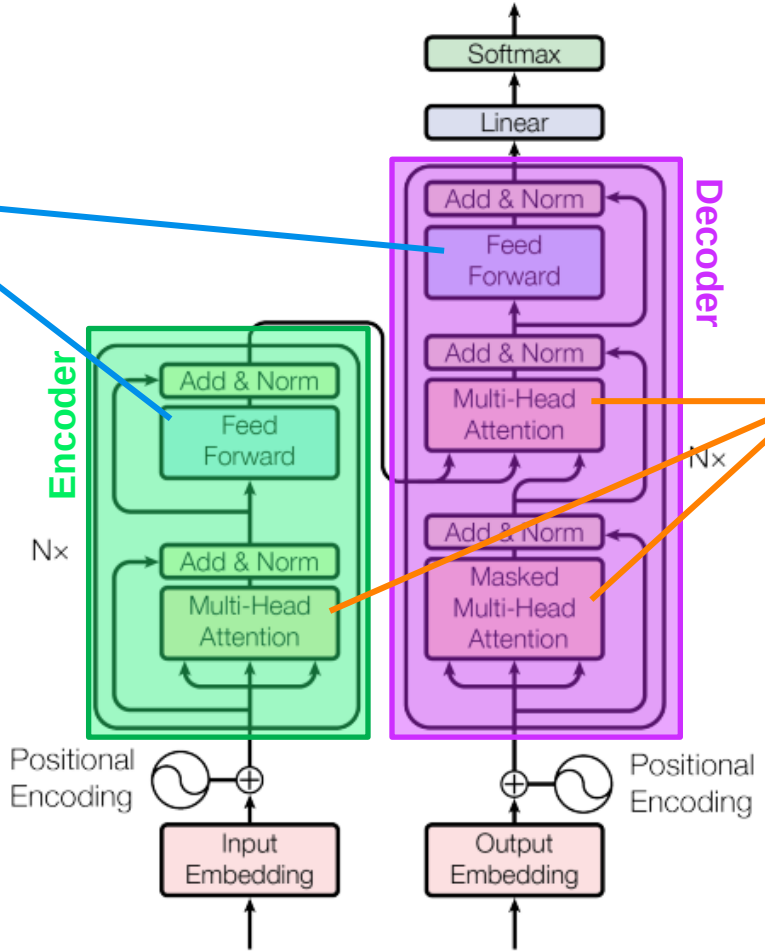
“Attention Is All You Need”

Next is Language Translation…

Feed Forward Networks

Simple network to ensure non-linearity

Captures more complex relationships between the input and output

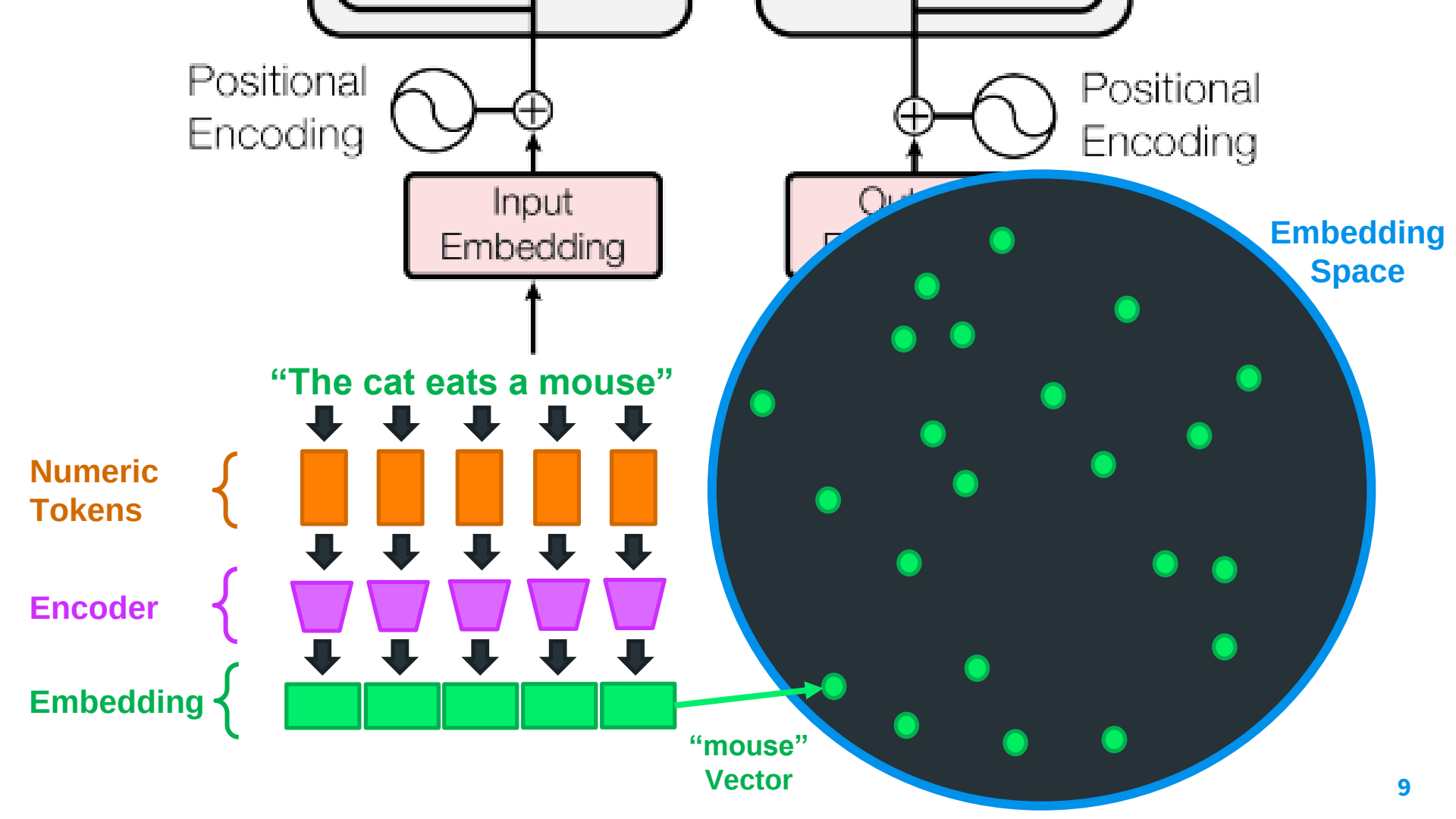


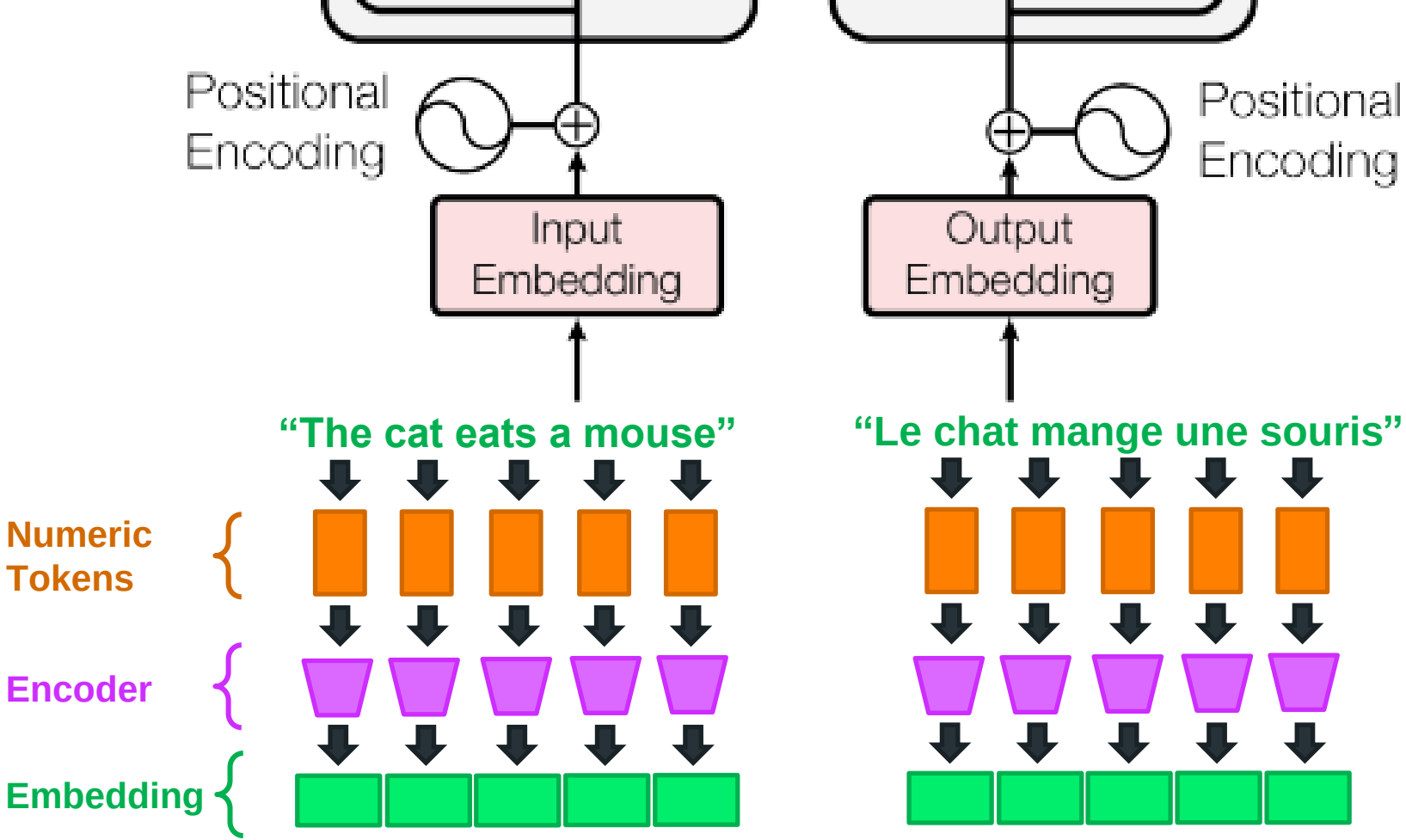
Some Attention Sprinkled Around

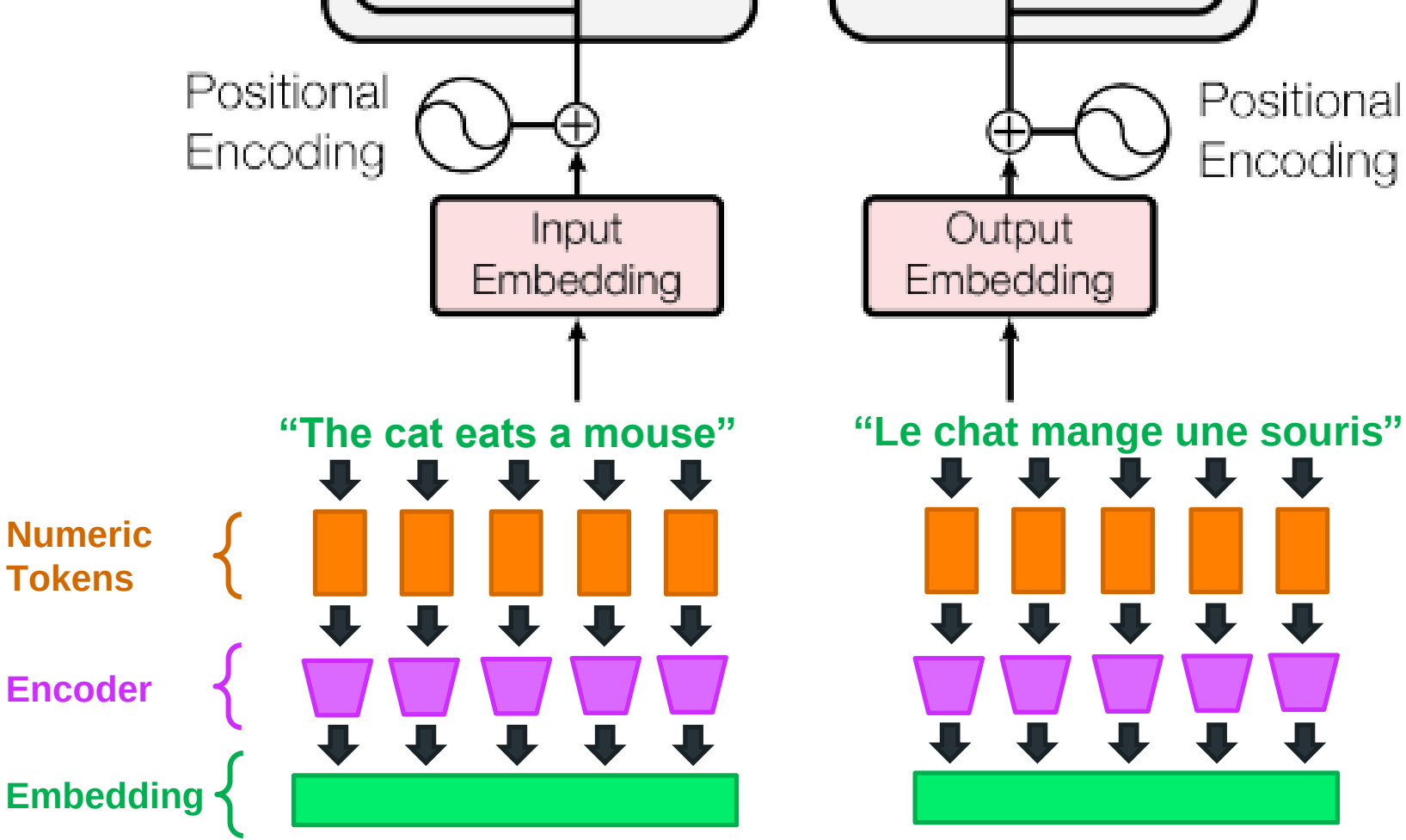
Can consider all portions of input sequence

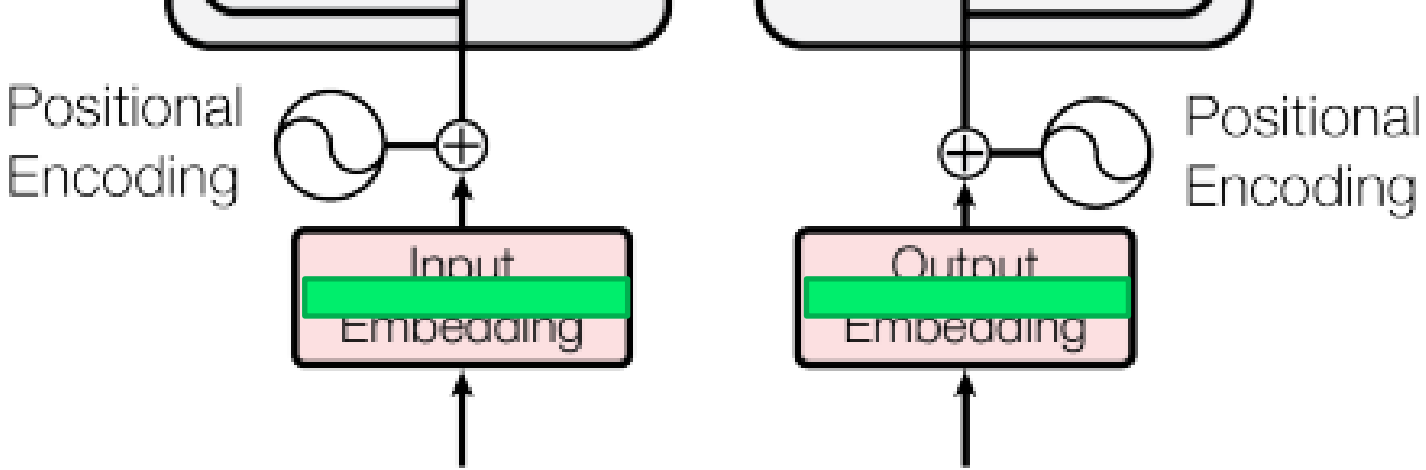
Can parallelize computation

We're going to look at training with a target sequence

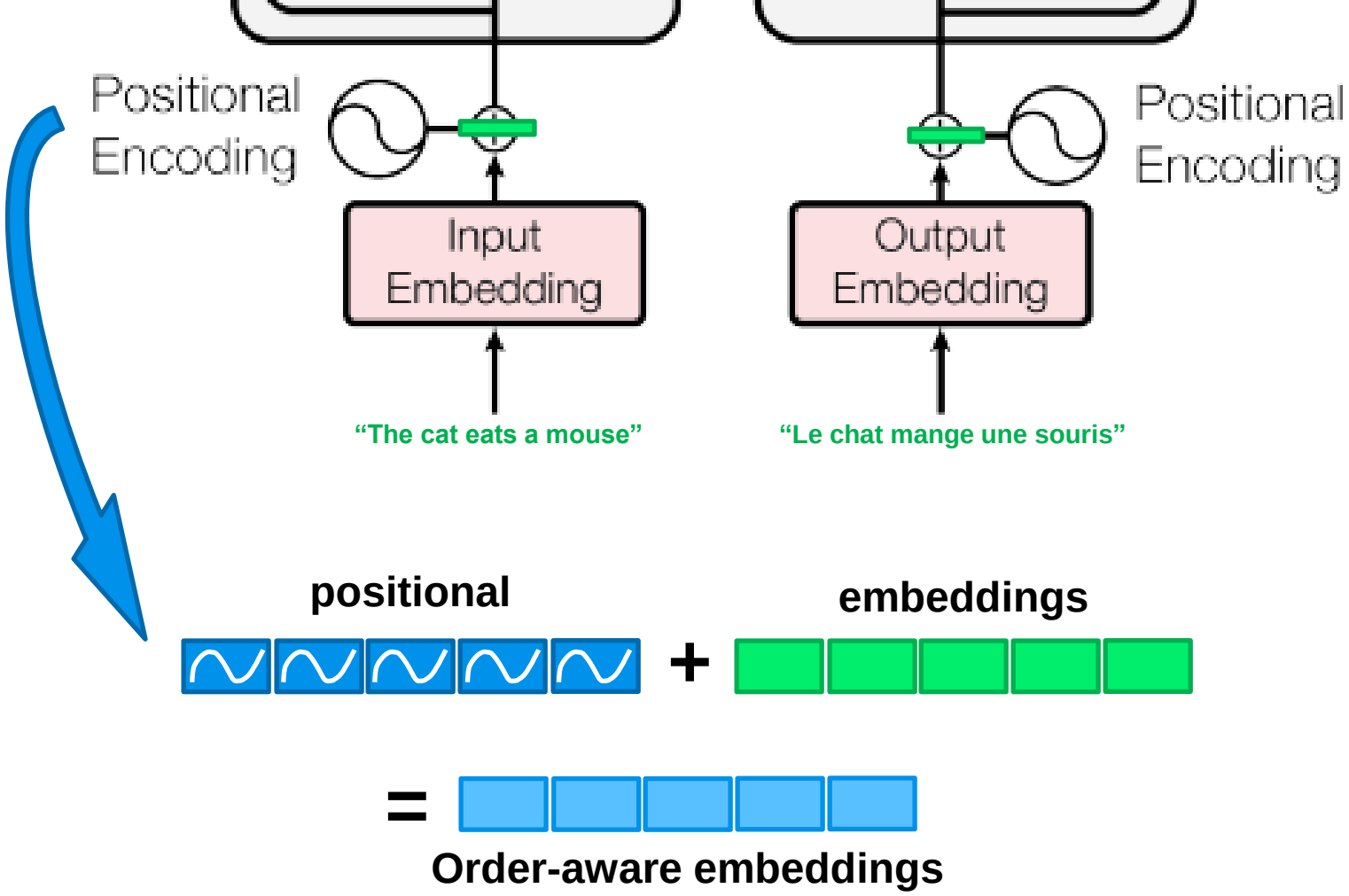

Positional Encoding
Input Embedding
Positional Encoding
Embedding Space
"The cat eats a mouse"
Numeric Tokens
Encoder
Embedding
"mouse" Vector

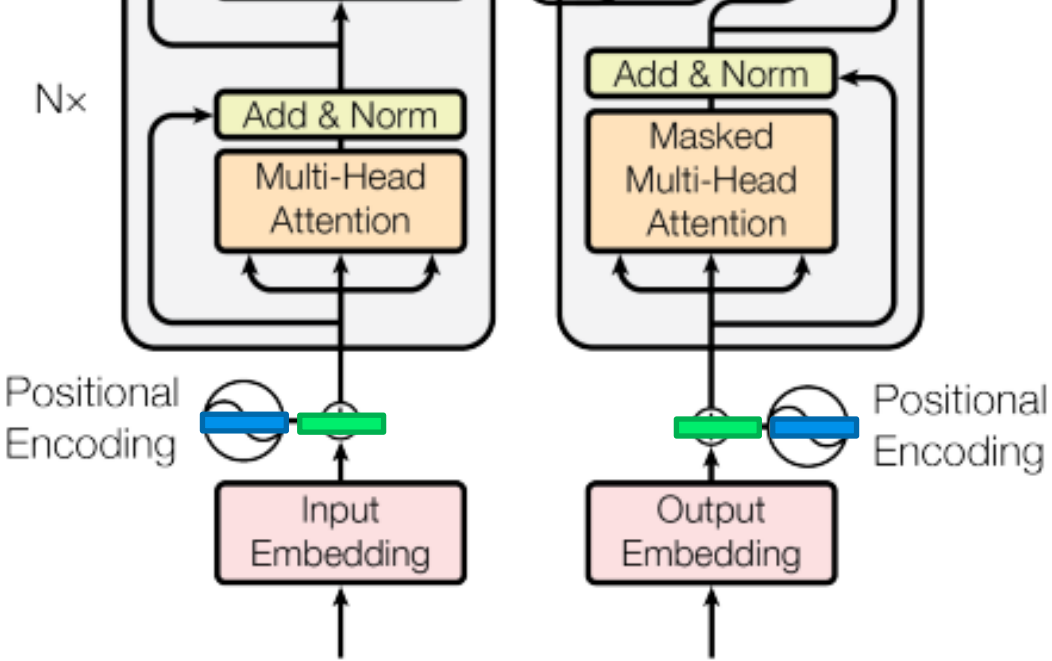

Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
"The cat eats a mouse"
"Le chat mange une souris"
Numeric Tokens
Encoder
Embedding

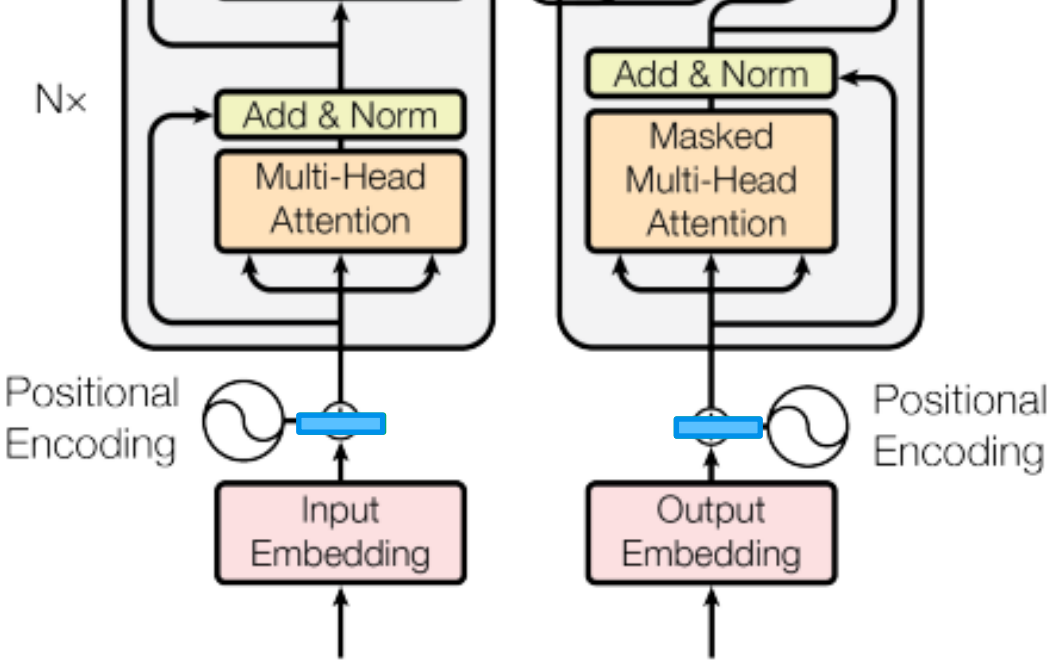

Positional Encoding
Input Embedding
Positional Encoding
Output Embedding
"The cat eats a mouse"
"Le chat mange une souris"
Numeric Tokens
Encoder
Embedding

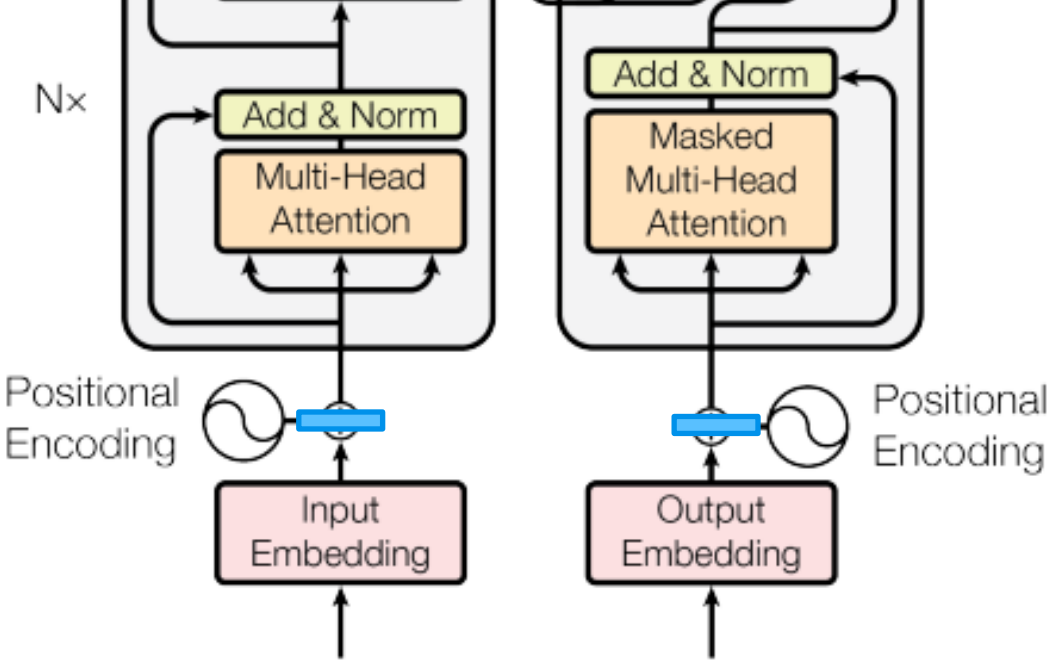


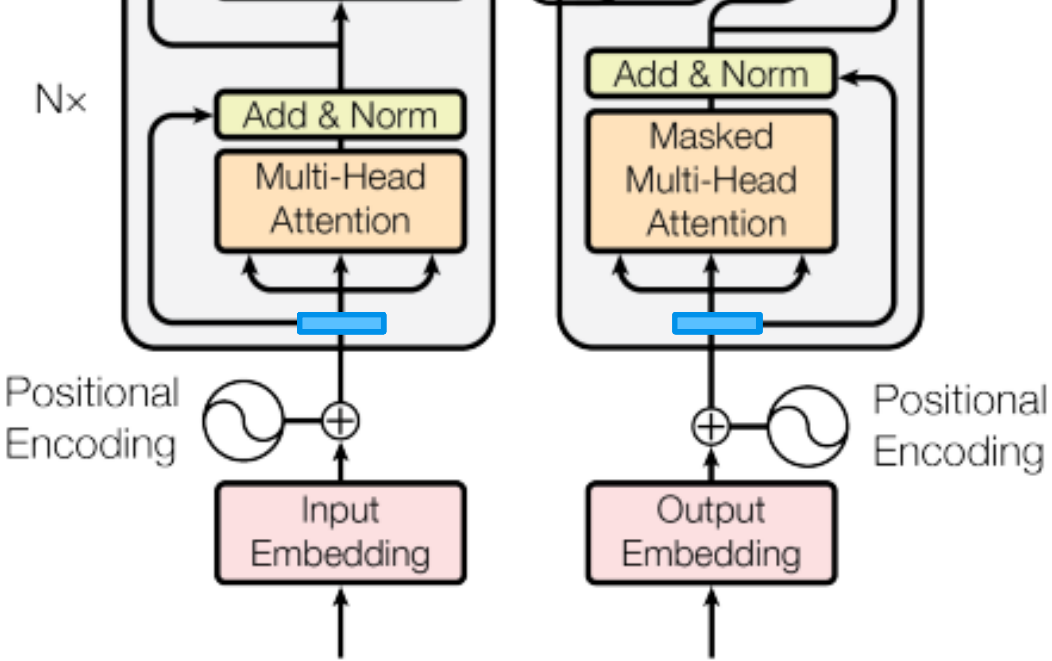
“The cat eats a mouse” **“Le chat mange une souris”**

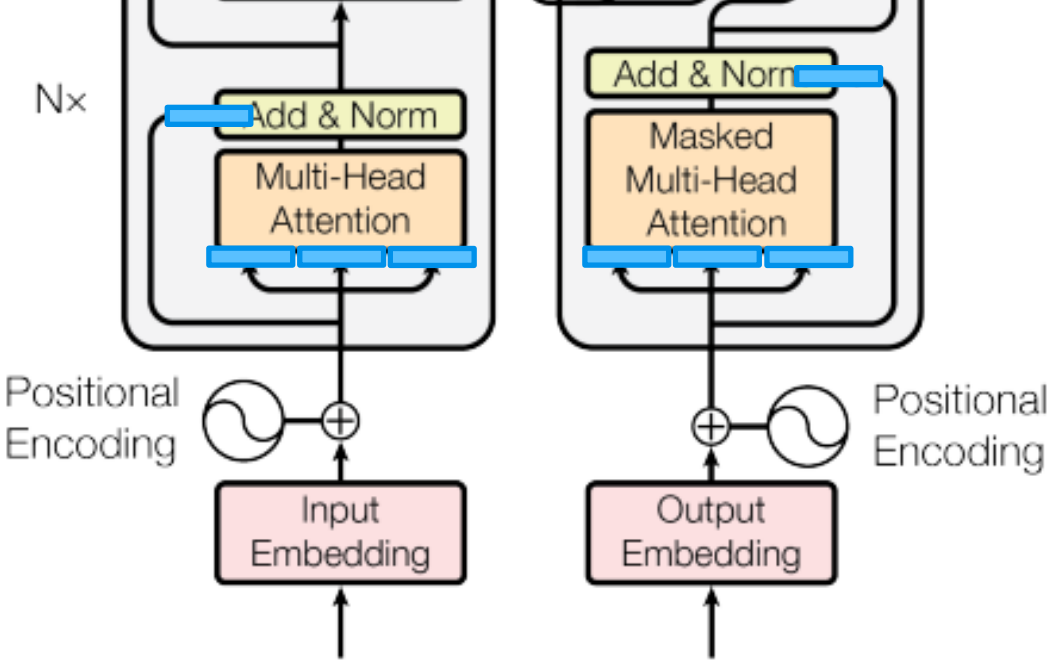

Positional Encoding
Input Embedding
Output Embedding
Positional Encoding
“The cat eats a mouse”
“Le chat mange une souris”
positional
embeddings
+
=
Order-aware embeddings


Nx
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
“The cat eats a mouse”
“Le chat mange une souris”

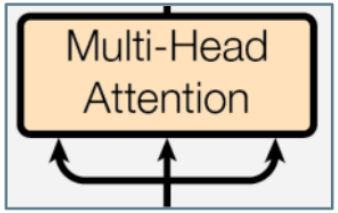

Nx
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
“The cat eats a mouse”
“Le chat mange une souris”


Nx
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
“The cat eats a mouse”
“Le chat mange une souris”


Nx
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
“The cat eats a mouse”
“Le chat mange une souris”

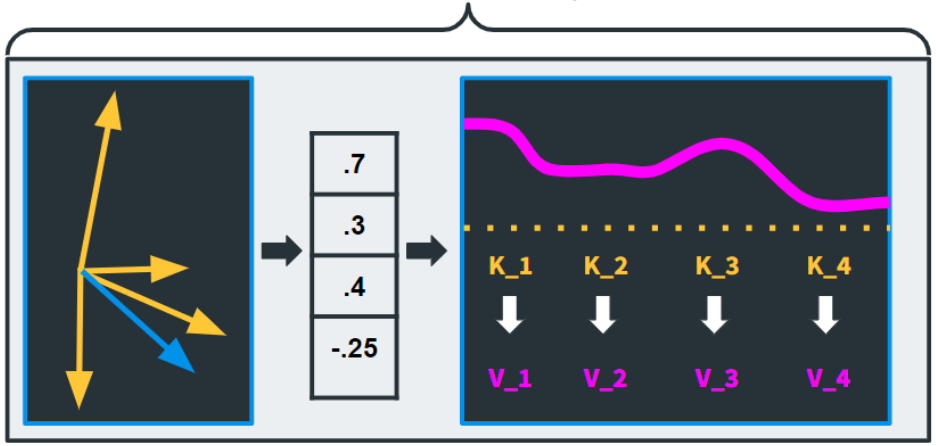

Nx
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
“The cat eats a mouse”
“Le chat mange une souris”

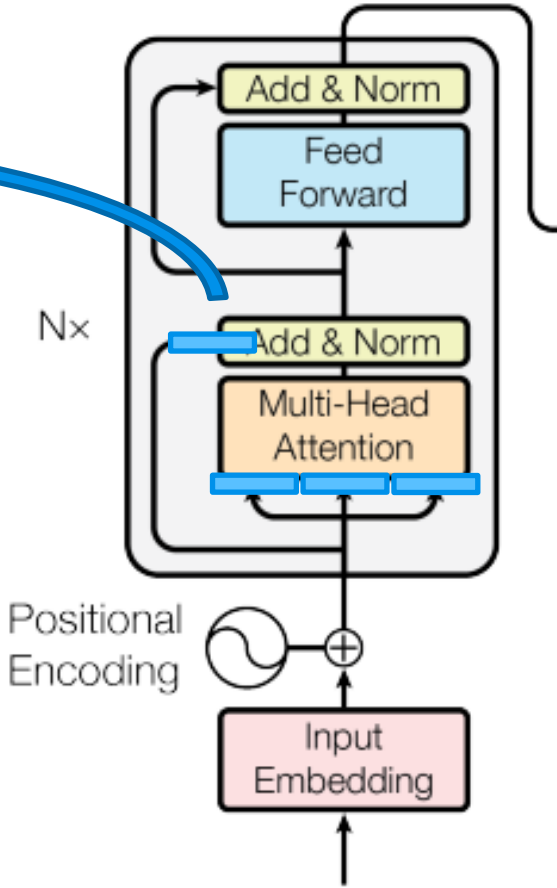
Using Attention!

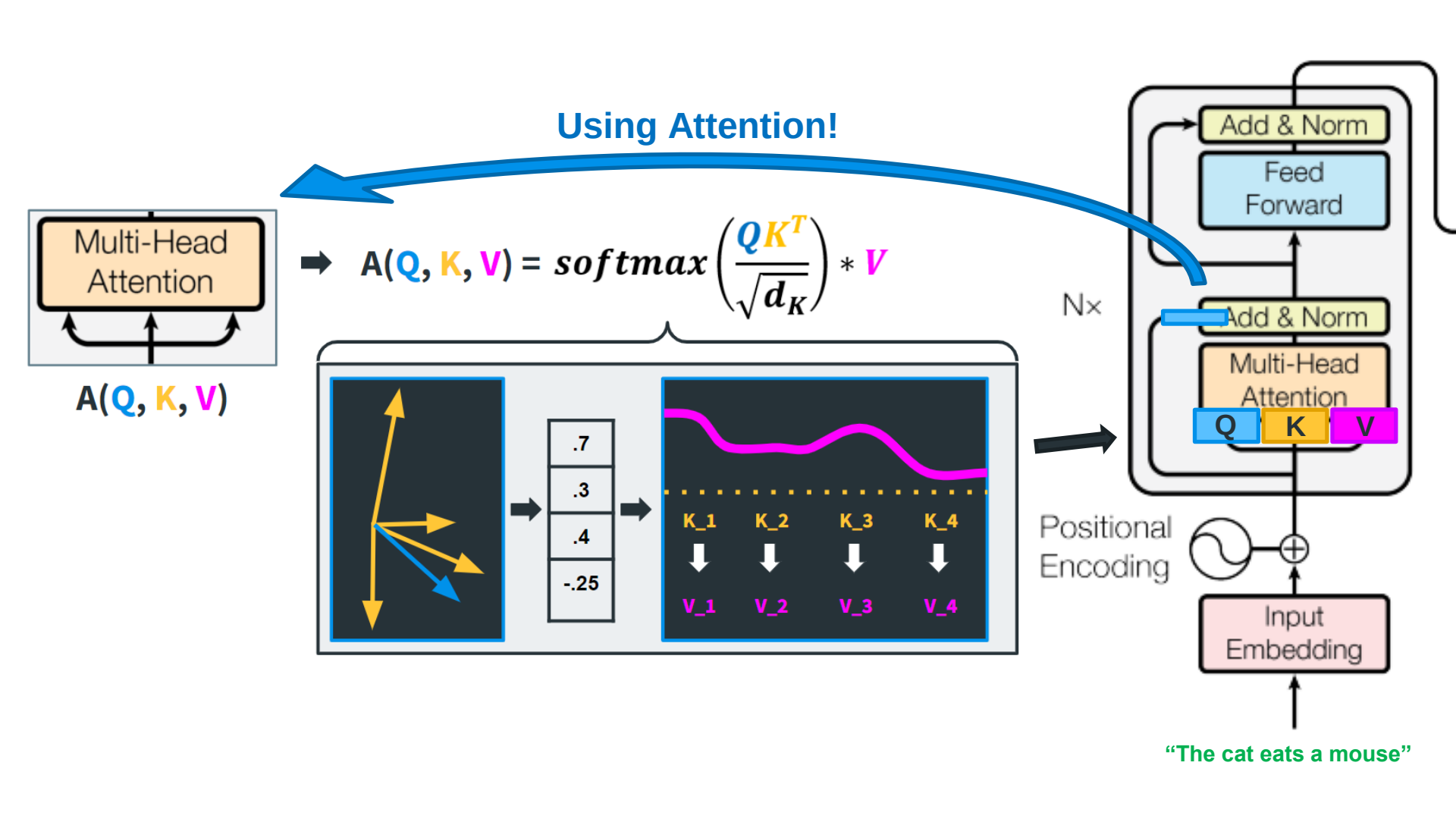


A(Q, K, V)

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





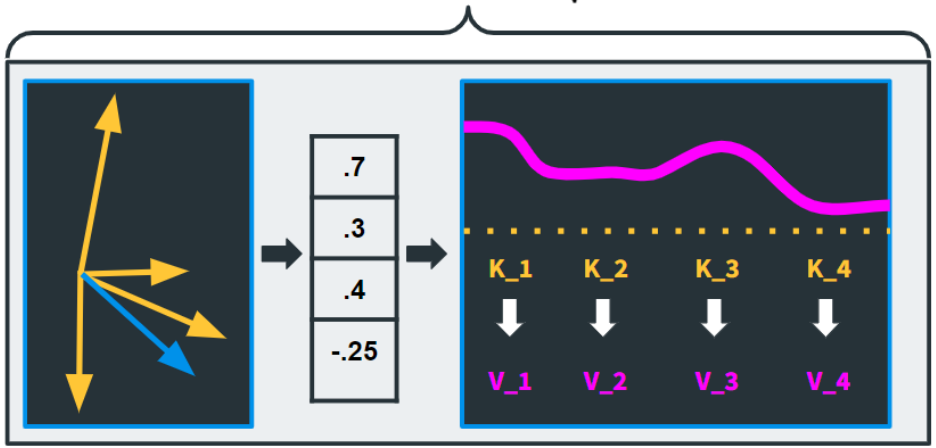

Using Attention!
Multi-Head Attention
A(Q, K, V)
A(Q, K, V) = $softmax\left(\frac{QK^T}{\sqrt{d_K}}\right) * V$
.7
.3
.4
-.25
K_1
K_2
K_3
K_4
V_1
V_2
V_3
V_4
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Q
K
V
Positional Encoding
Input Embedding
"The cat eats a mouse"

Using Attention!

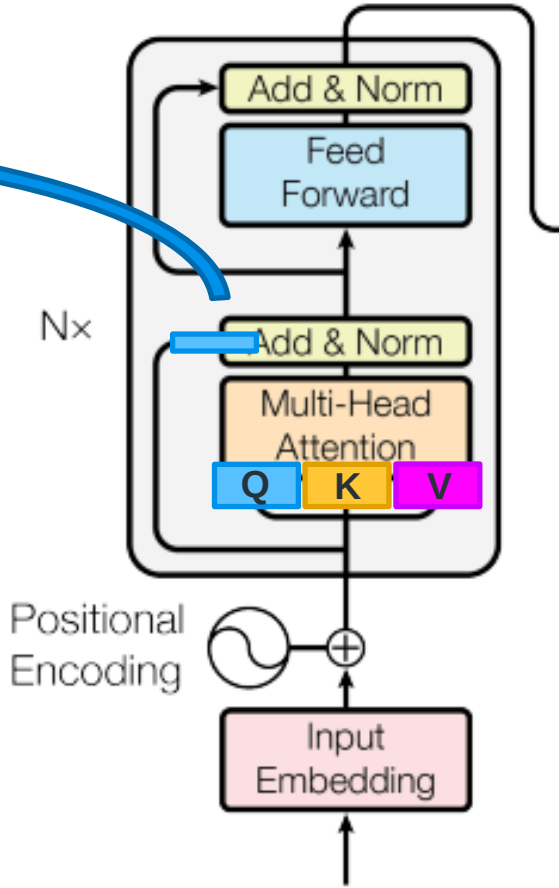
Multi-Head Attention

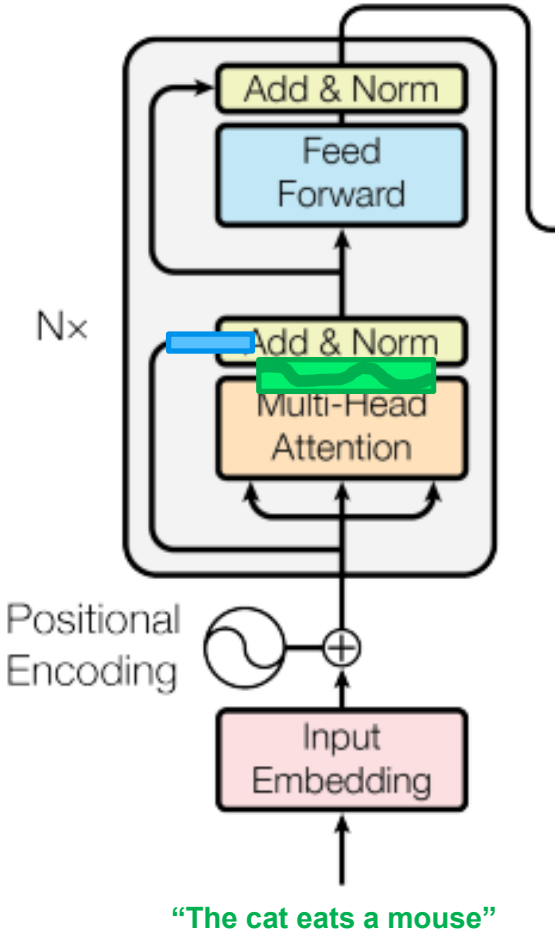
A(Q, K, V)

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



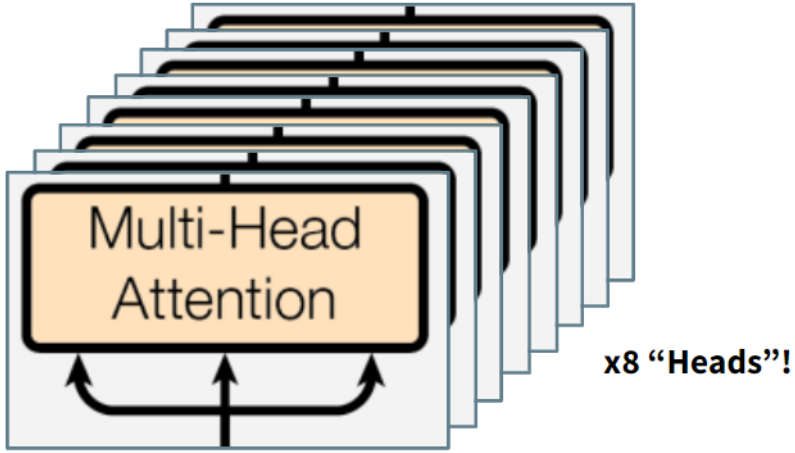
Attention-Filtered Embedding =




Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
“The cat eats a mouse”

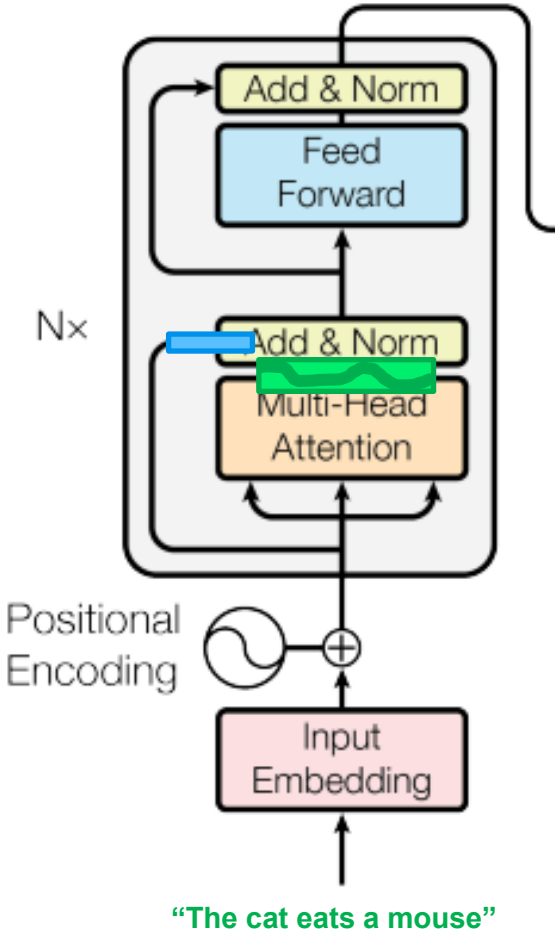
Multi-Headed Attention

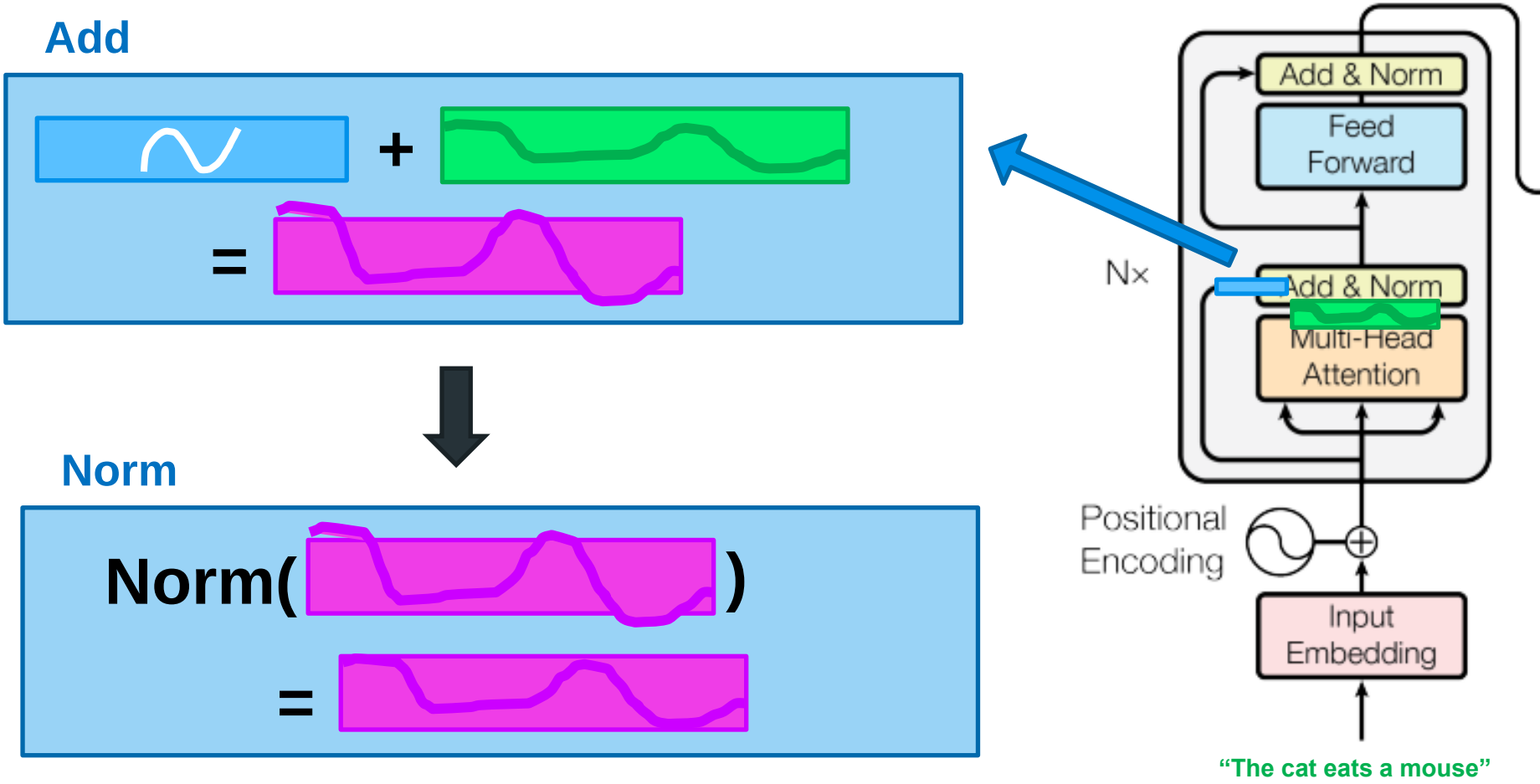
$A(Q, K, V) =$



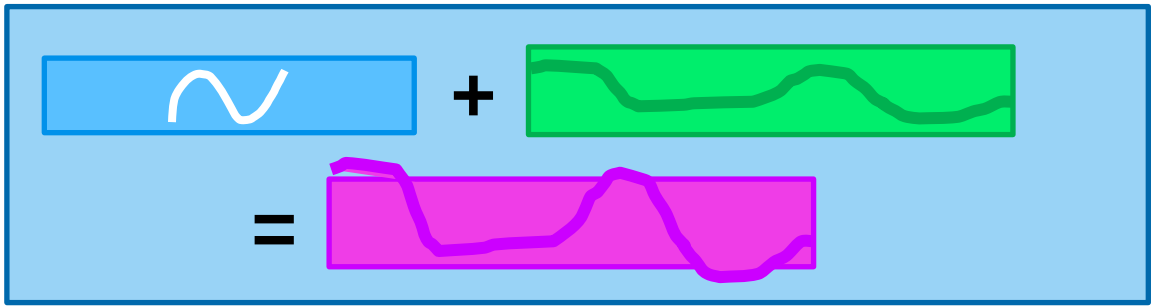
x8 “Heads”!

Multiple heads capture more abstract features

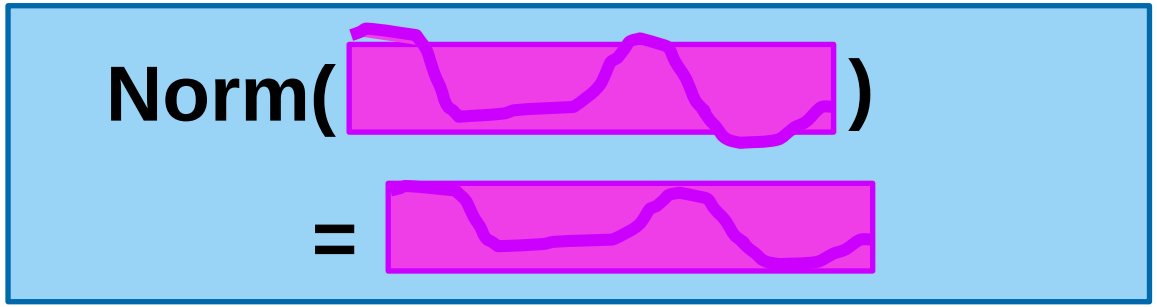


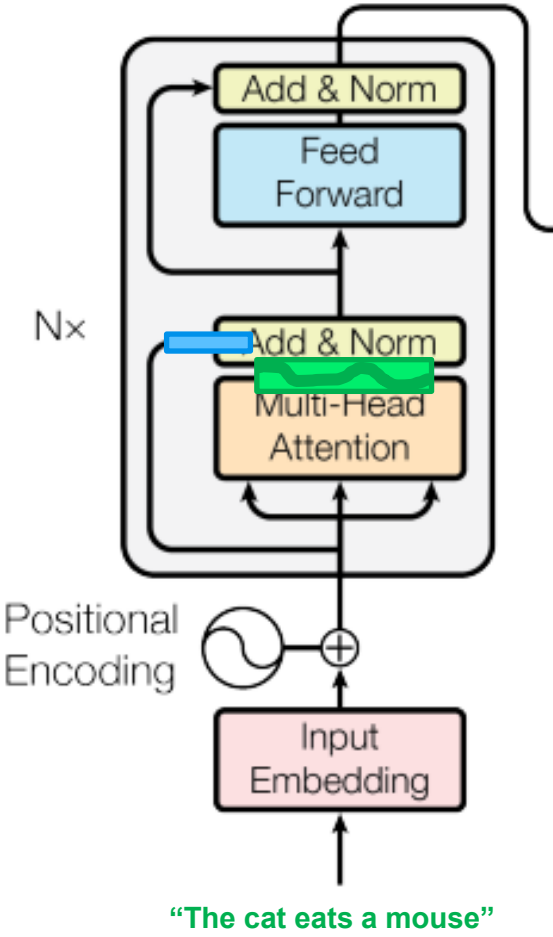

Add
+
=
Norm
Norm(
)
=
Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
"The cat eats a mouse"

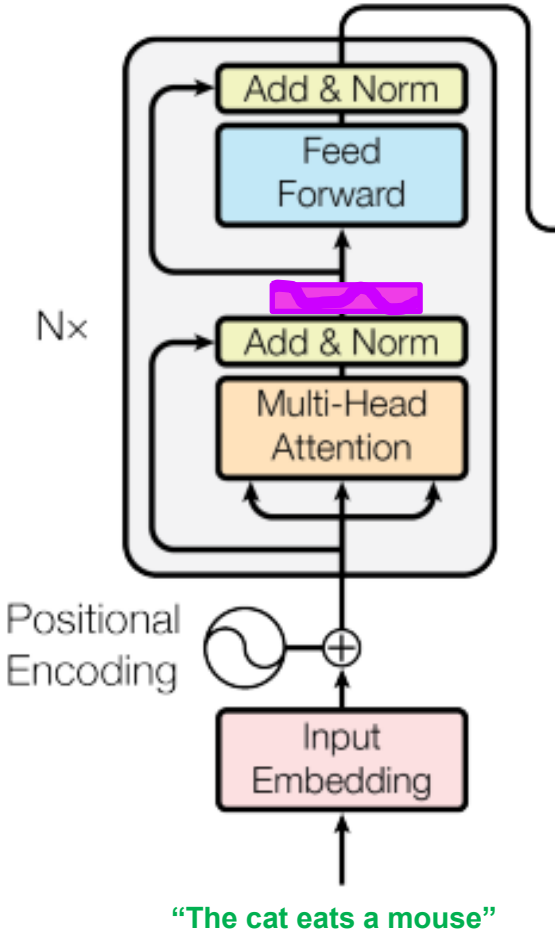
Add

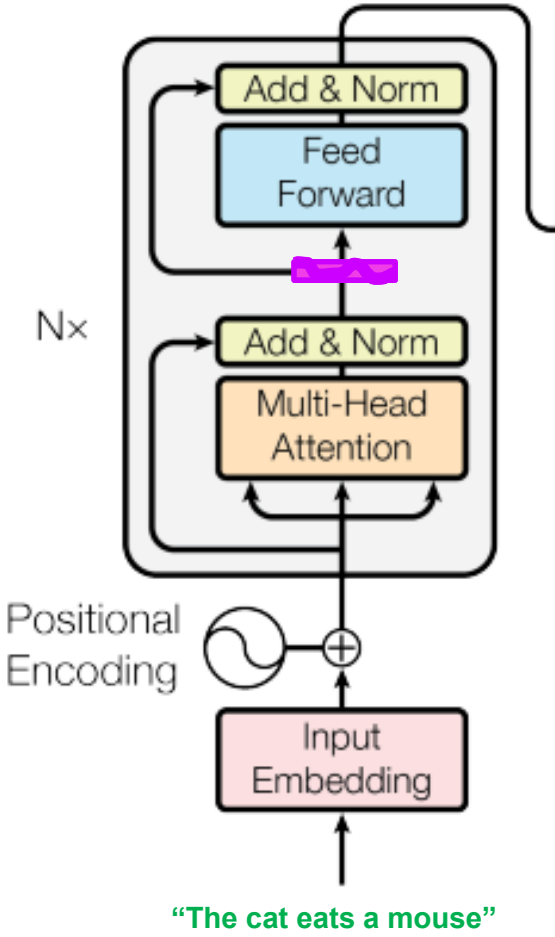


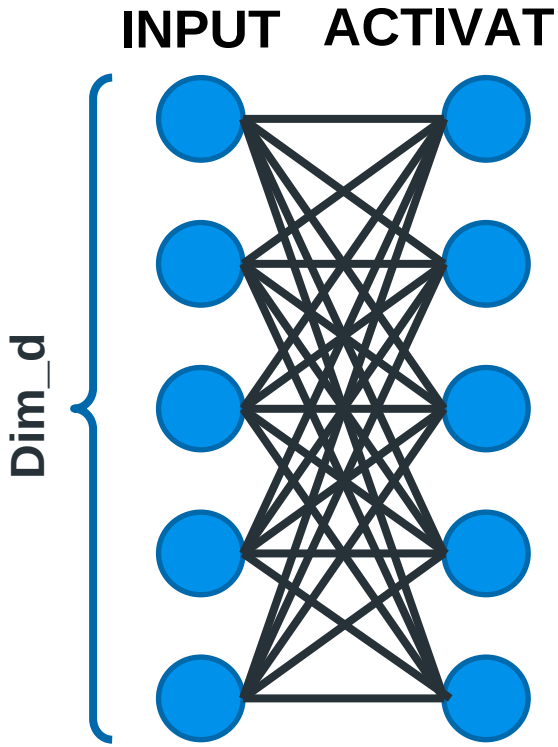
Norm

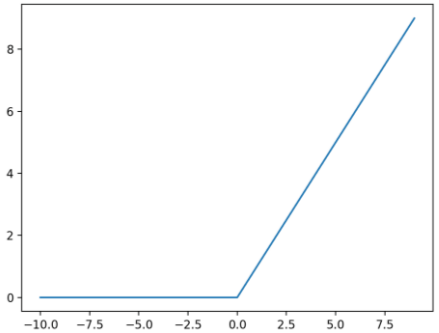

Norm(
)
=


Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
"The cat eats a mouse"


Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
“The cat eats a mouse”

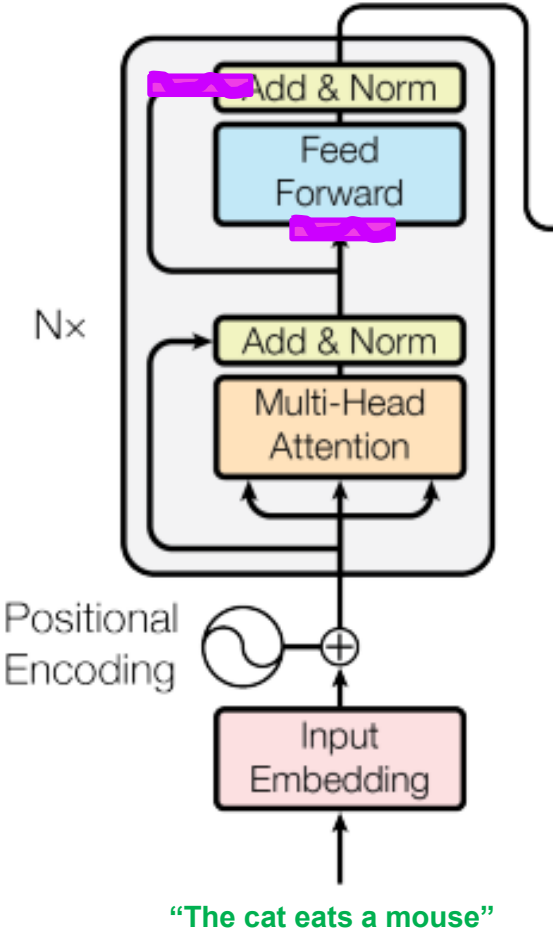

Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
“The cat eats a mouse”

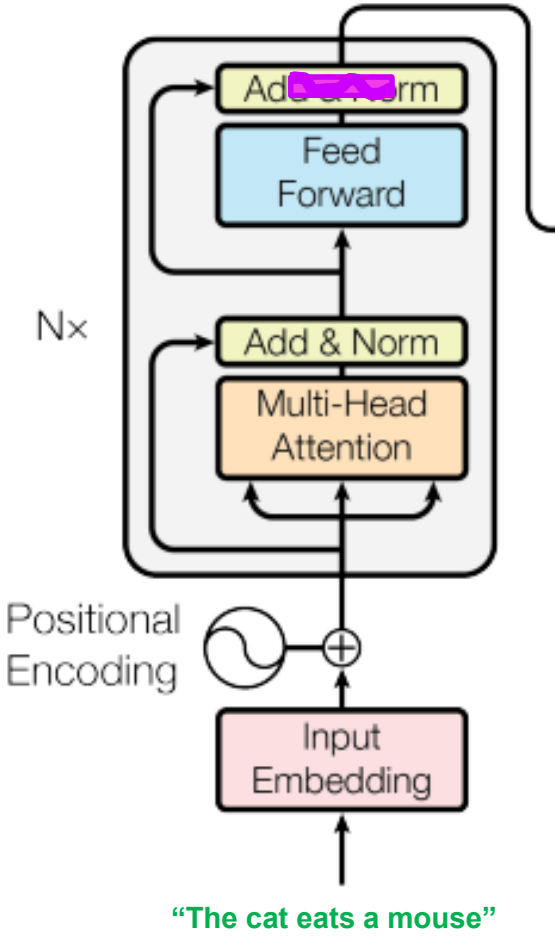


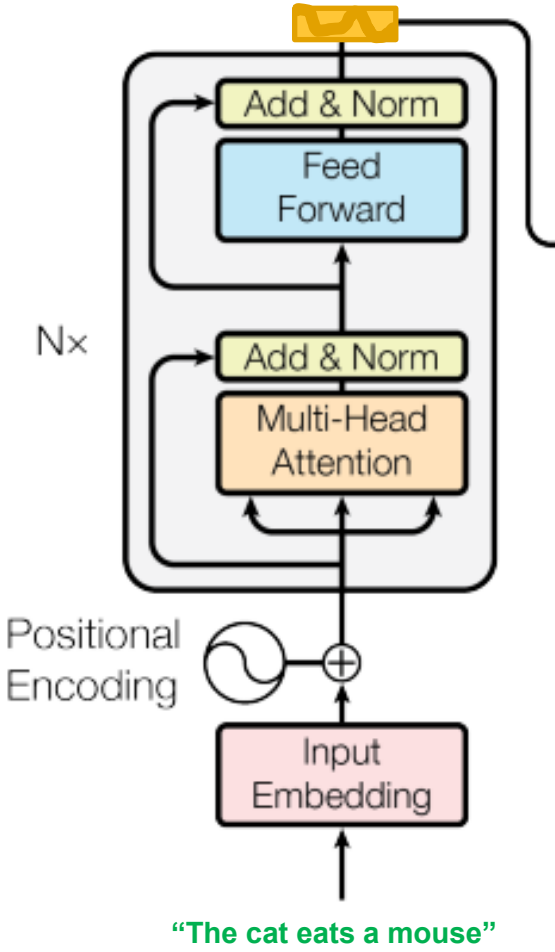


ReLU

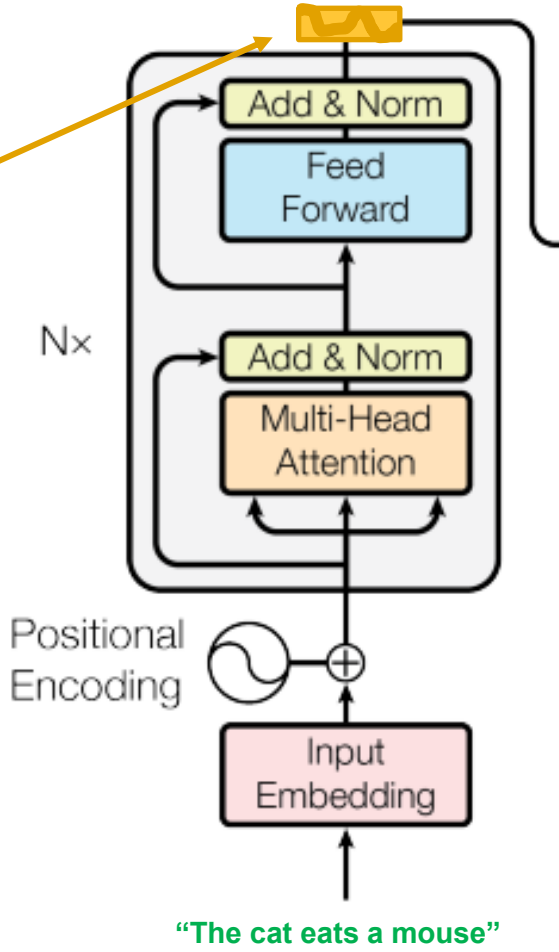
Non-Linearity
Simple
Mitigate Vanishing




Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
“The cat eats a mouse”

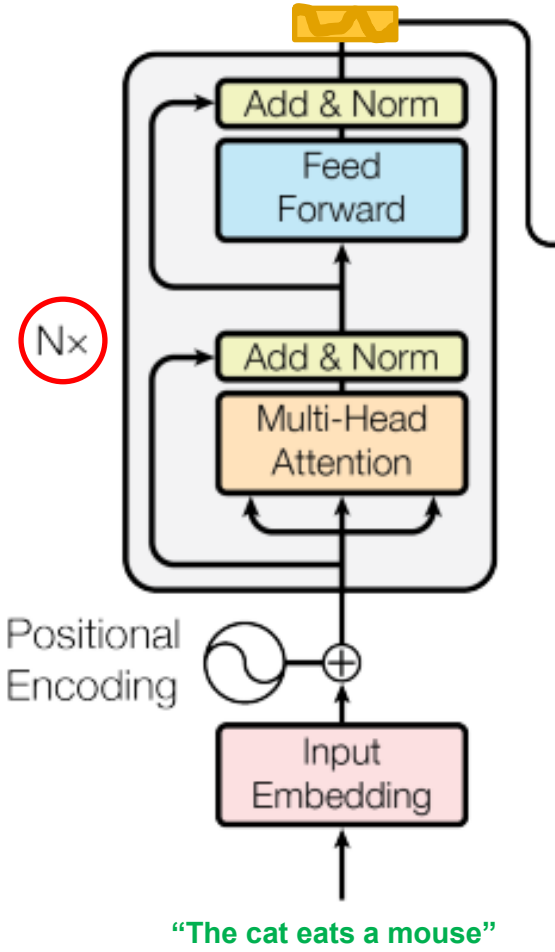

Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
“The cat eats a mouse”

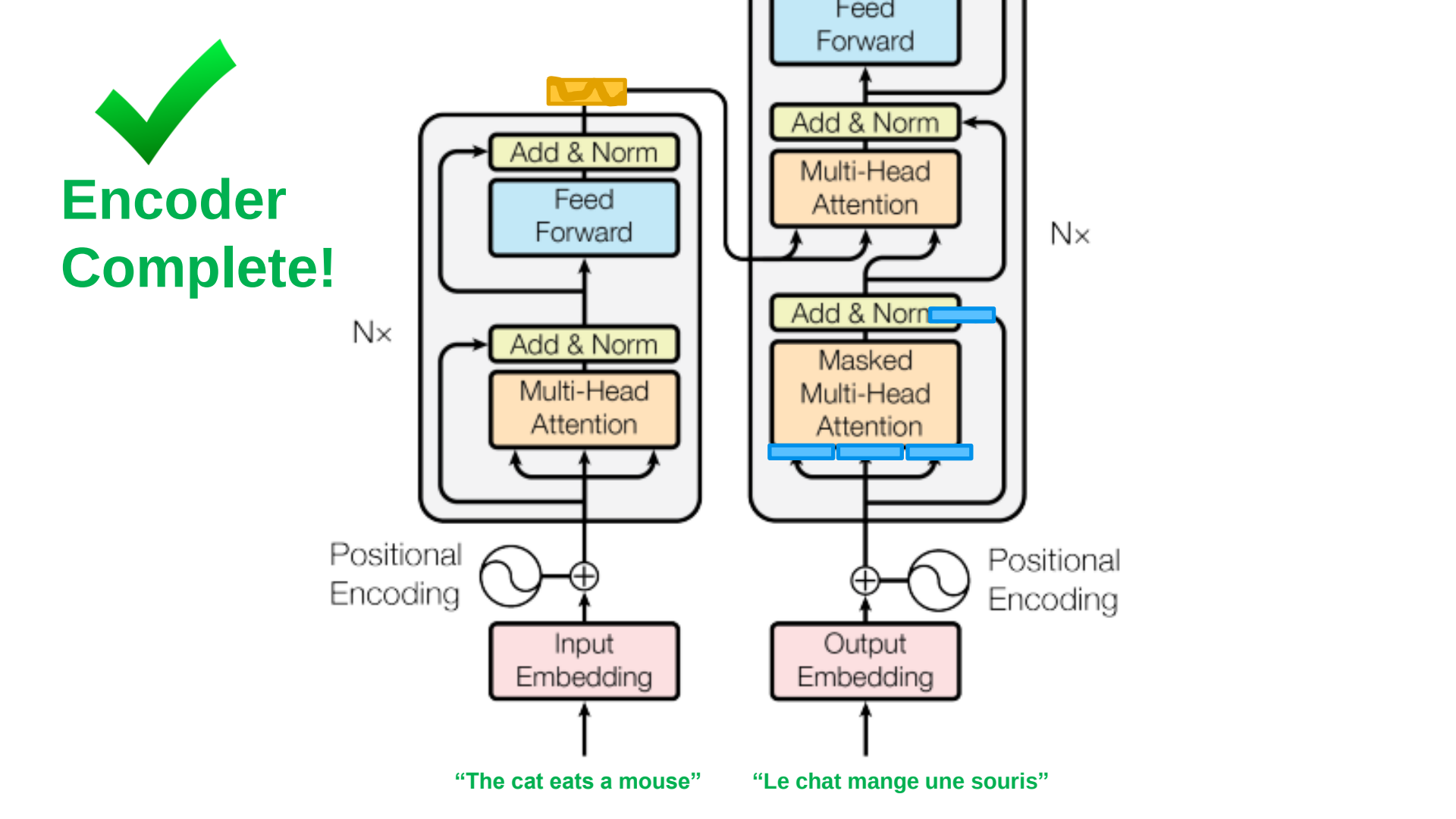
Refined Sentence Embeddings

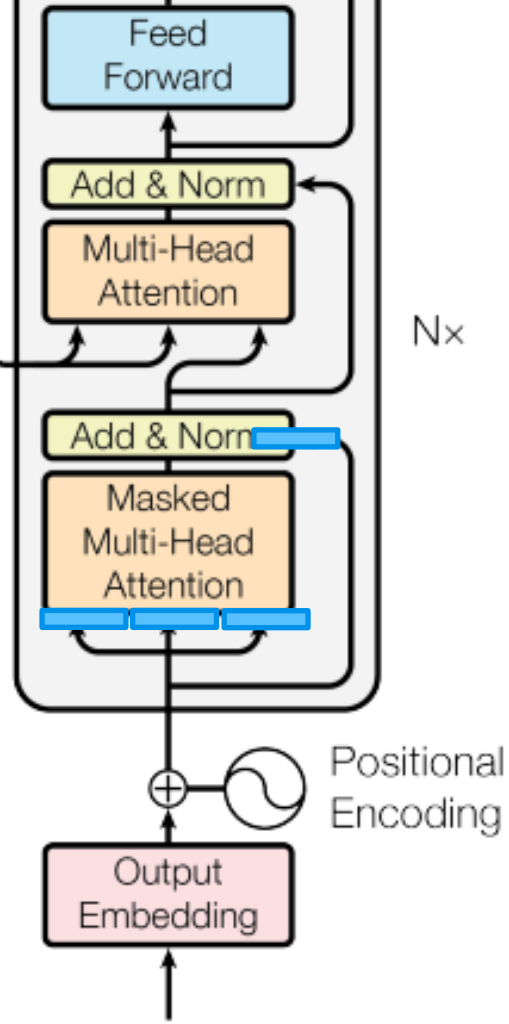


Multiple Encoders?!

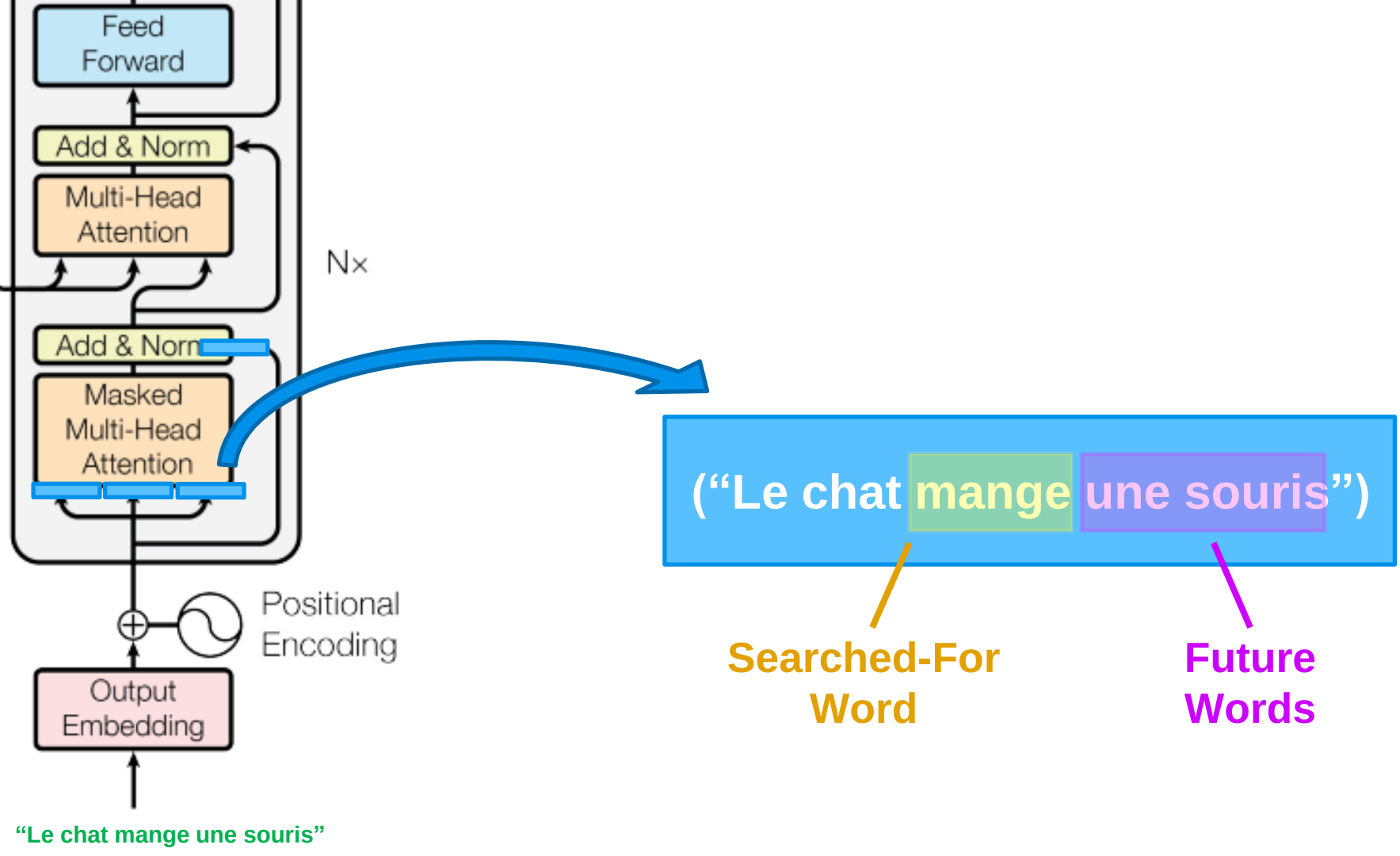
There are 6…

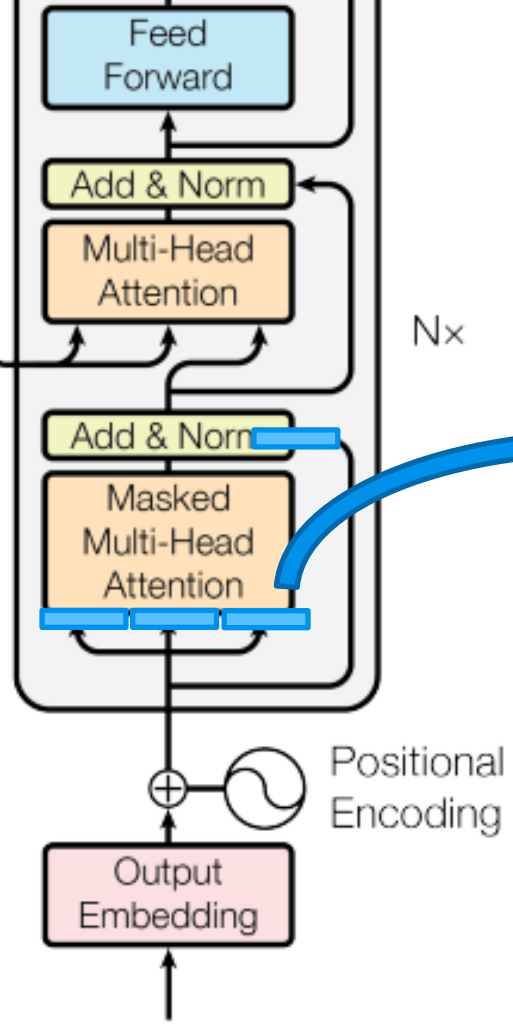



Encoder Complete!
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
"The cat eats a mouse"
Nx
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Positional Encoding
Output Embedding
"Le chat mange une souris"



“Le chat mange une souris”

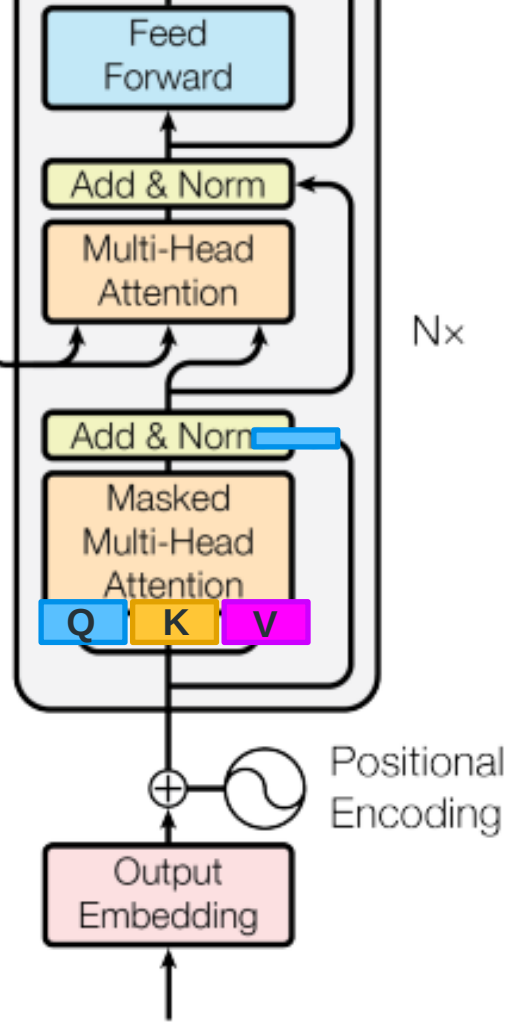

Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
Output Embedding
“Le chat mange une souris”
(“Le chat mange une souris”)
Searched-For Word
Future Words



Let's add a mask…

("Le chat [redacted]")

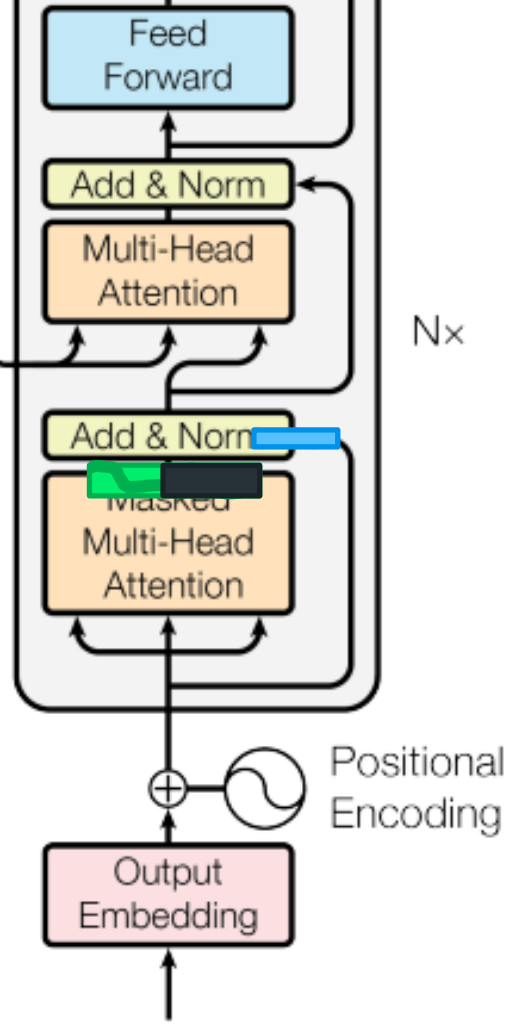
- **Prevents Looking Ahead**
- **Efficiency**



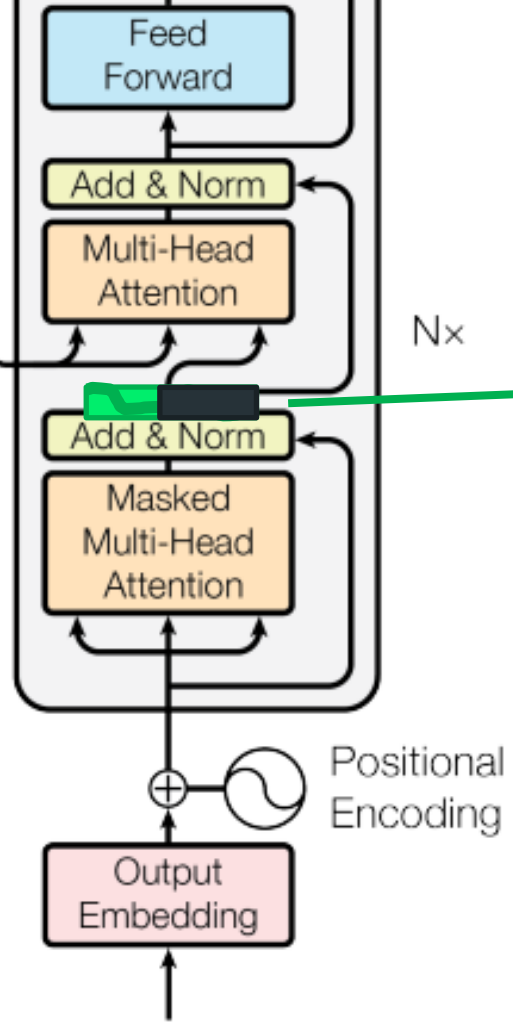
N×

“Le chat mange une souris”

(“Le chat [redacted]”)

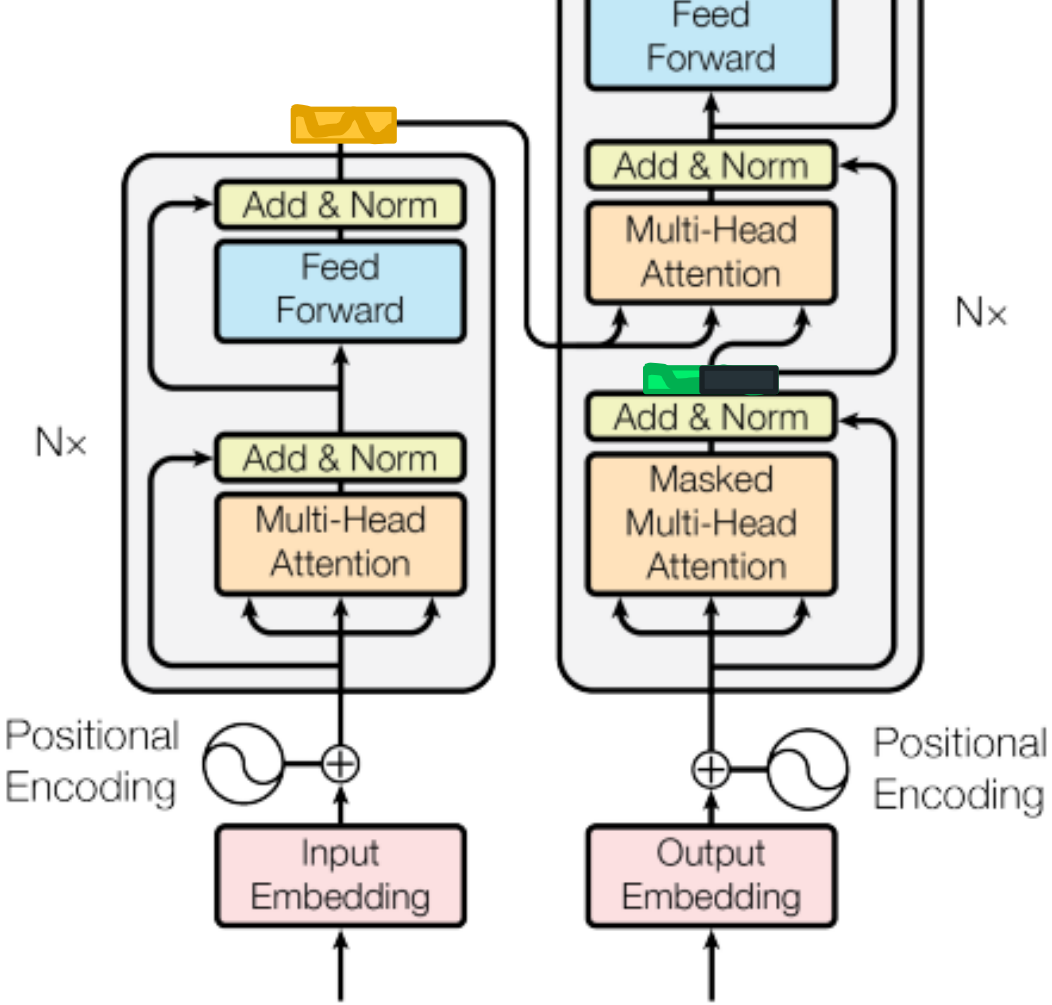


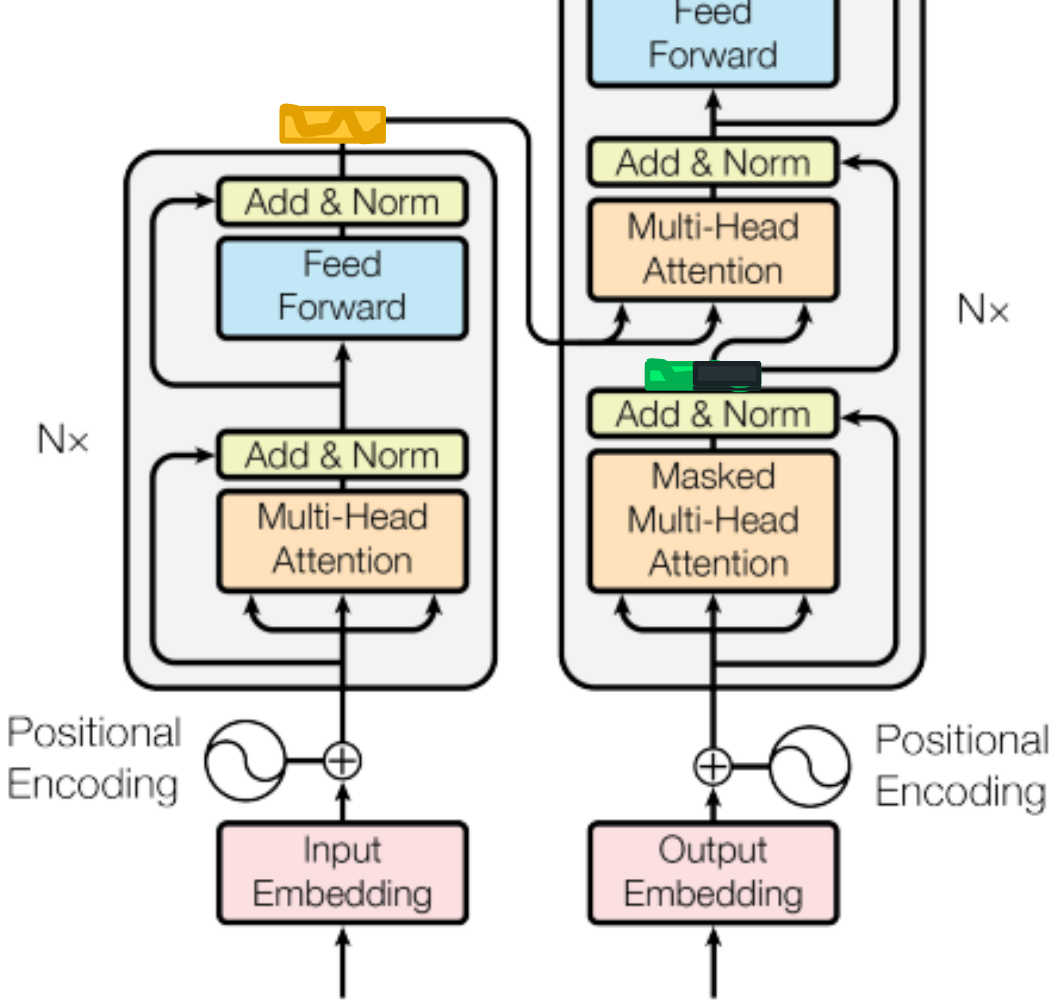
“Le chat mange une souris”

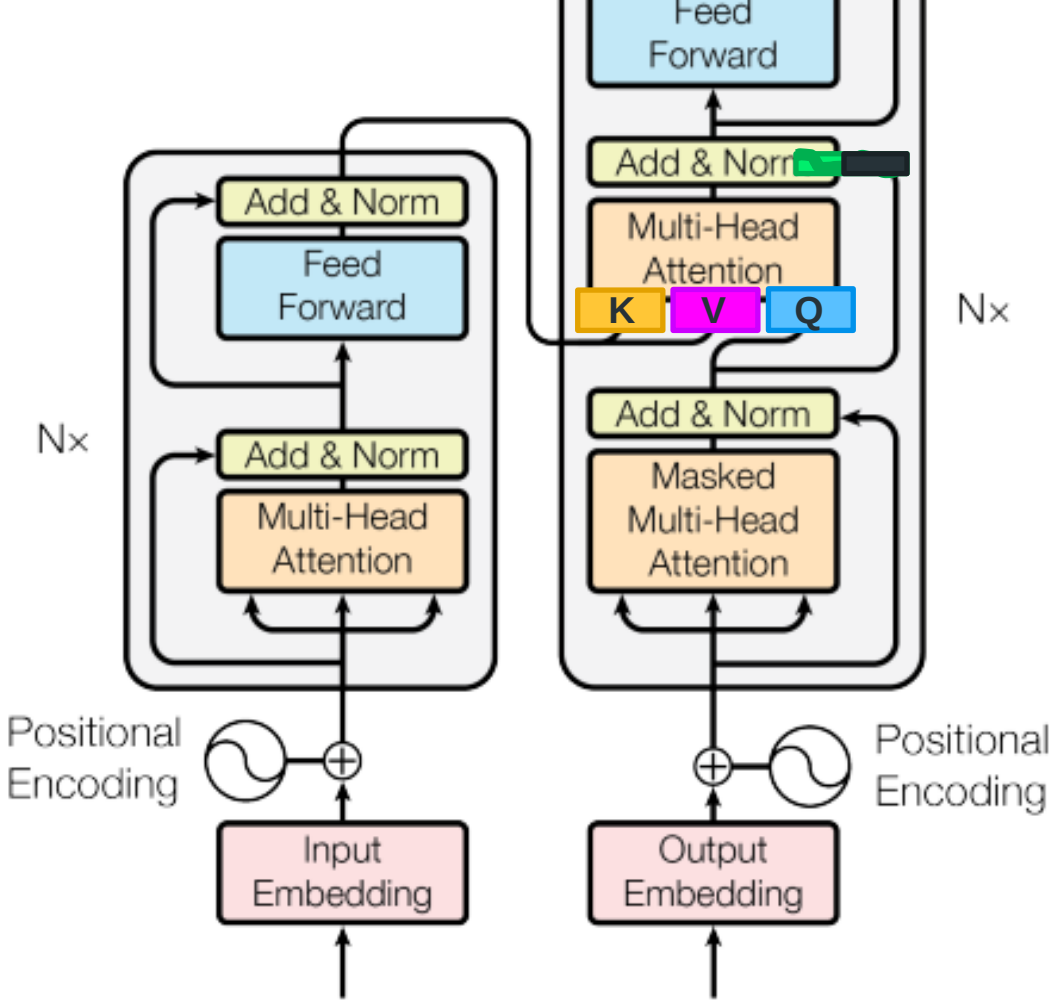


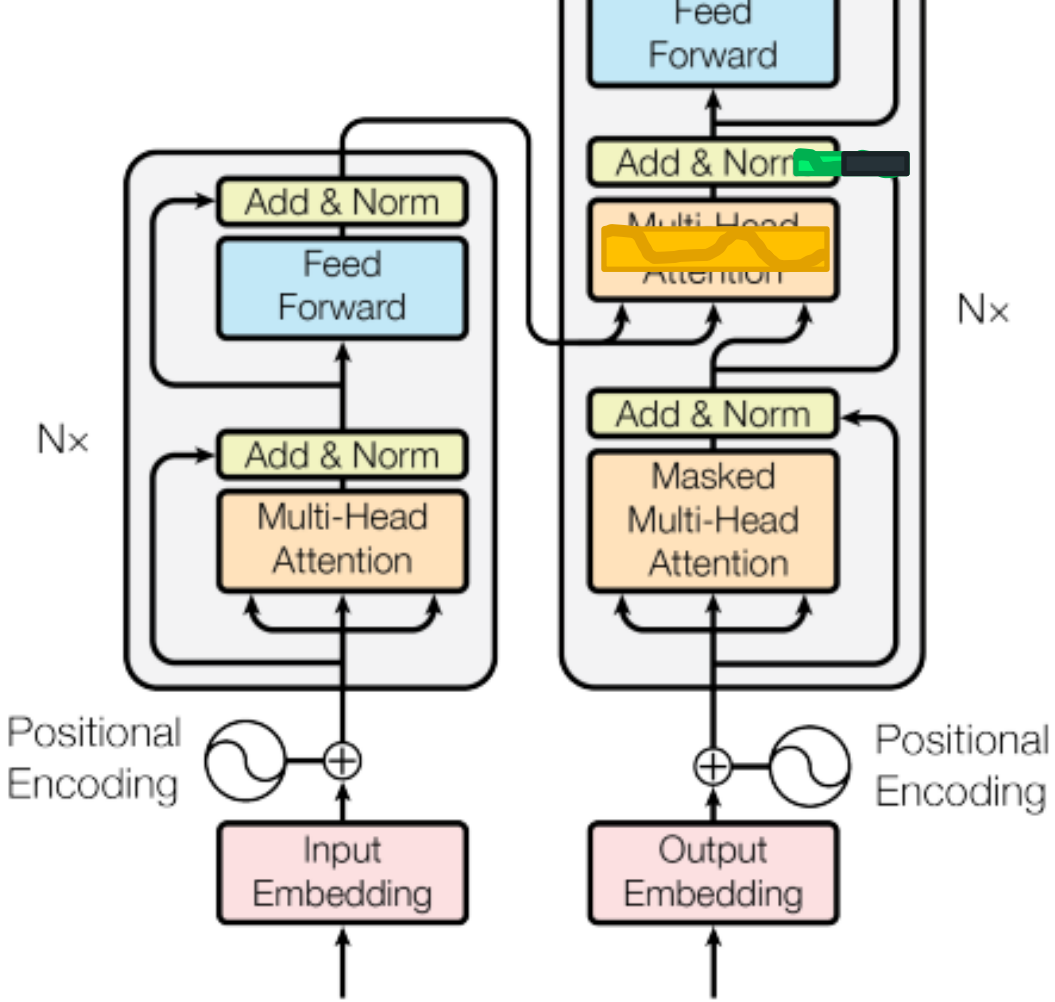
Output Sentence Embeddings

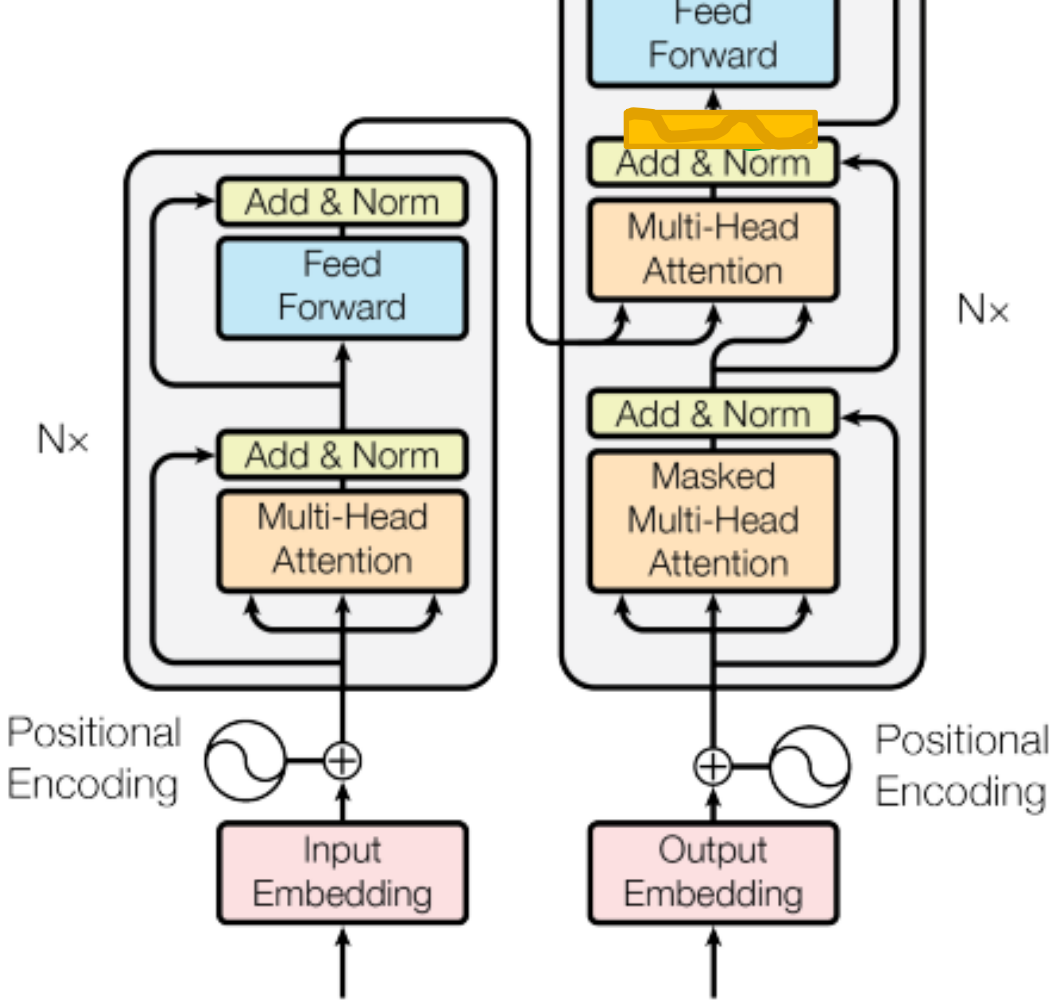
“Le chat mange une souris”

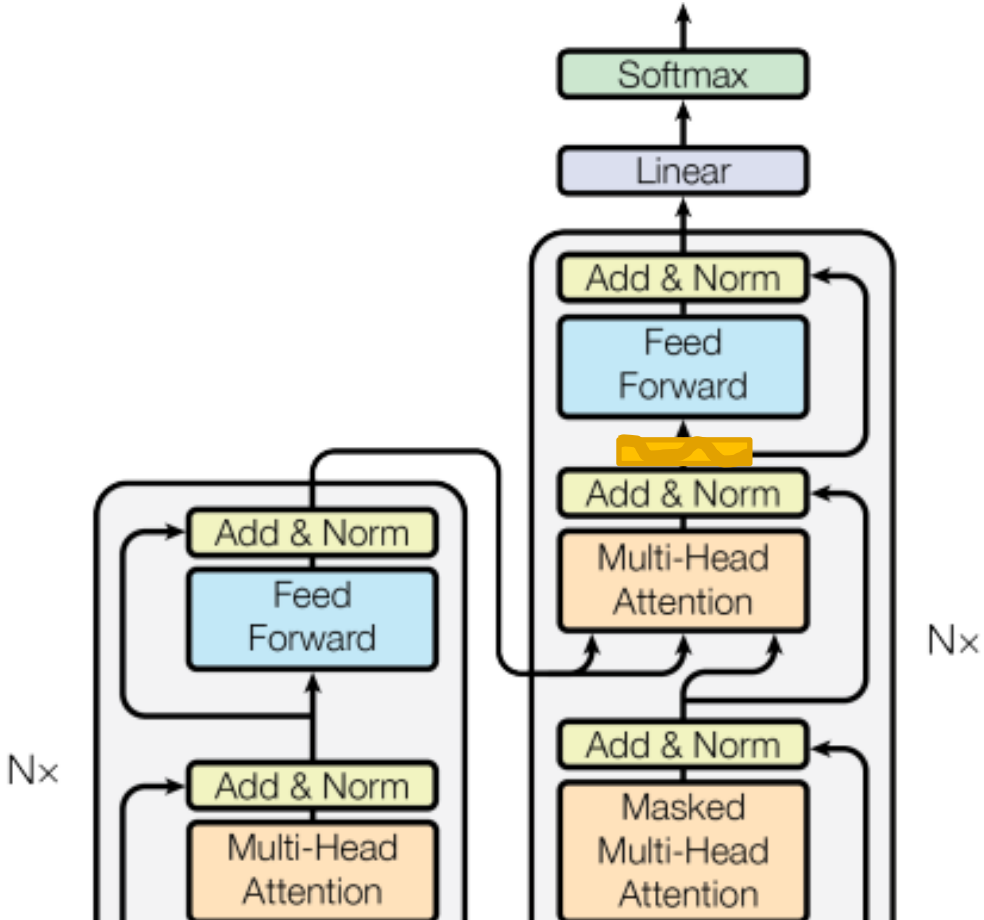

Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
“The cat eats a mouse”
“Le chat mange une souris”

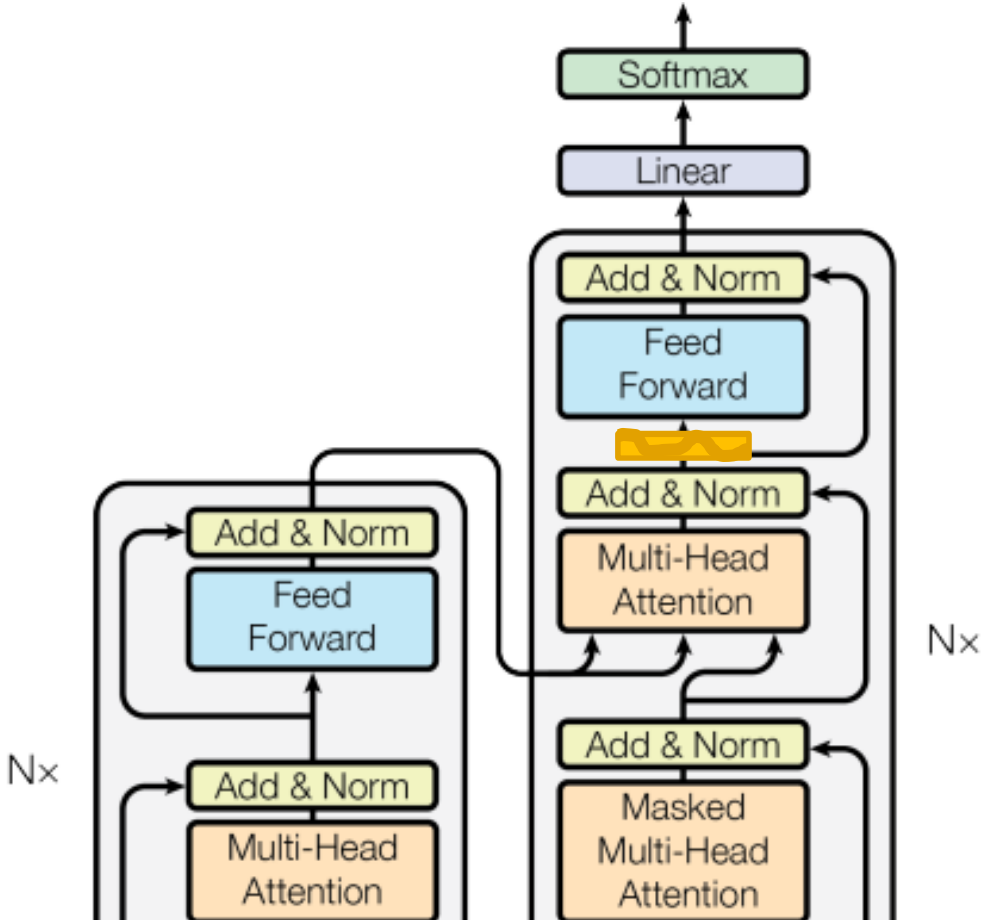

Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
“The cat eats a mouse”
“Le chat mange une souris”

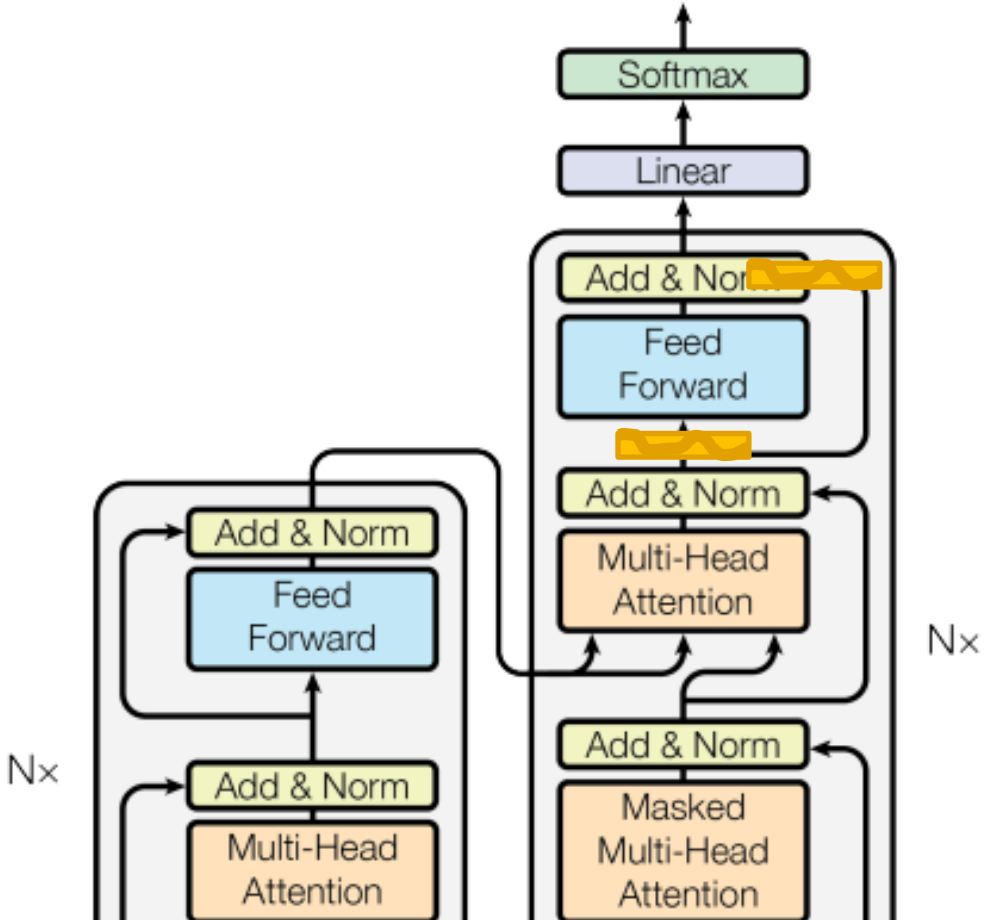

Feed Forward
Add & Norm
Multi-Head Attention
K
V
Q
Add & Norm
Masked Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
“The cat eats a mouse”
“Le chat mange une souris”

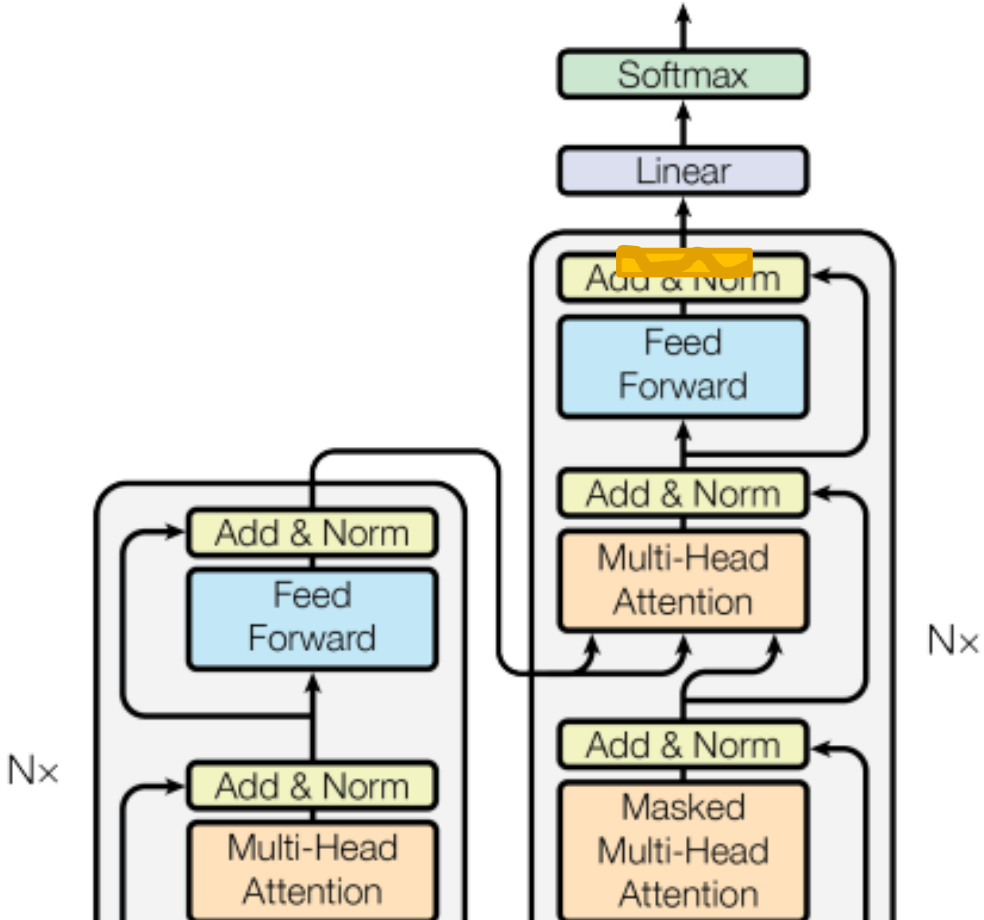

Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
"The cat eats a mouse"
"Le chat mange une souris"

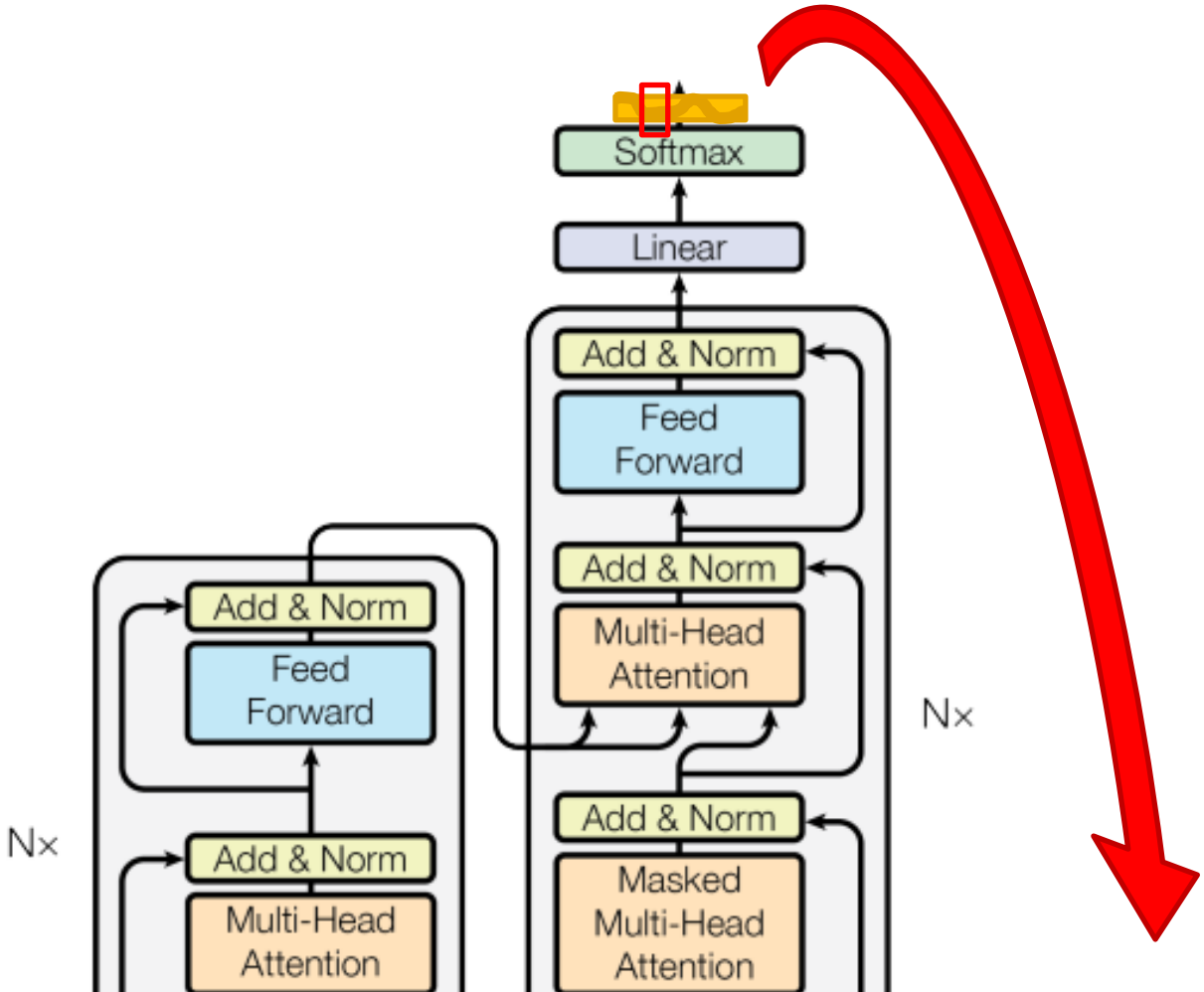

Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
“The cat eats a mouse”
“Le chat mange une souris”

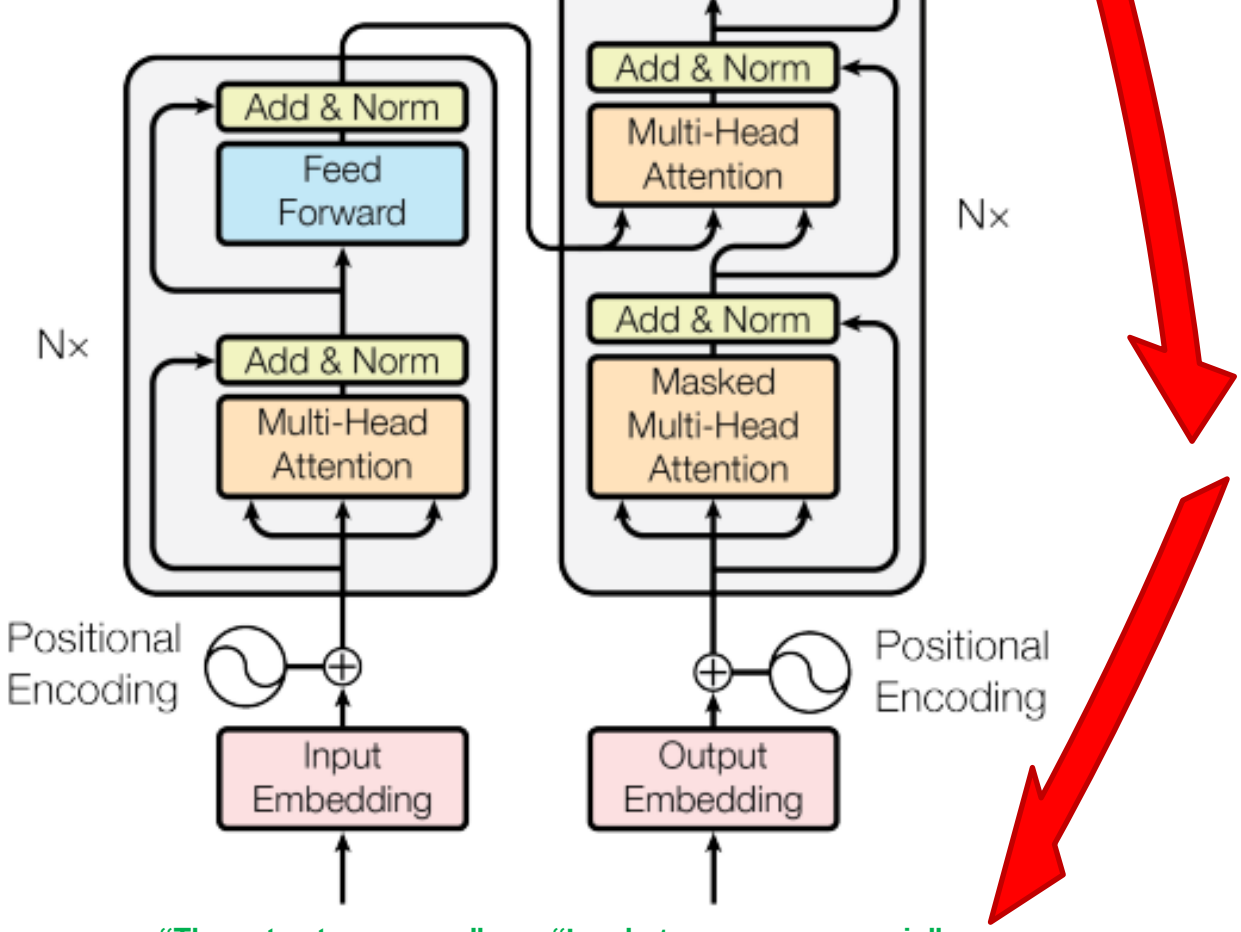

Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx


Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx


Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx

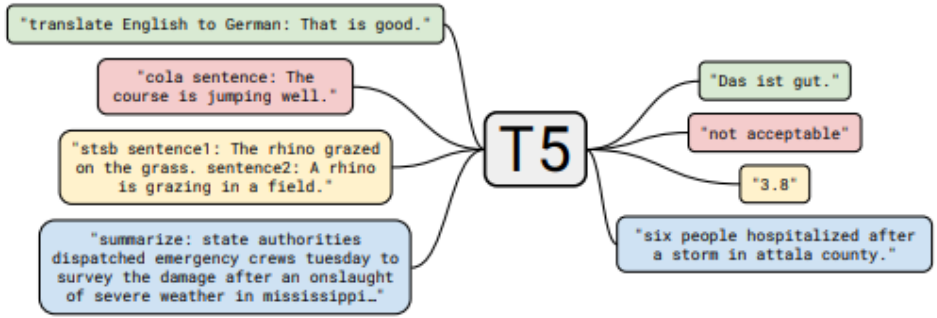

Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx


Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx


Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Input Embedding
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
Output Embedding
“The cat eats a mouse”
“Le chat mange une souris”

English to French - Model

- Used a pre-trained transformer from **HuggingFace** 🤗
- Available data
  - Paired English to French
  - Tokenizer
  - Fine-Tuning Data



Text-**T**o-**T**ext **T**ransfer **T**ransformer **(T5)**

English to French - Results

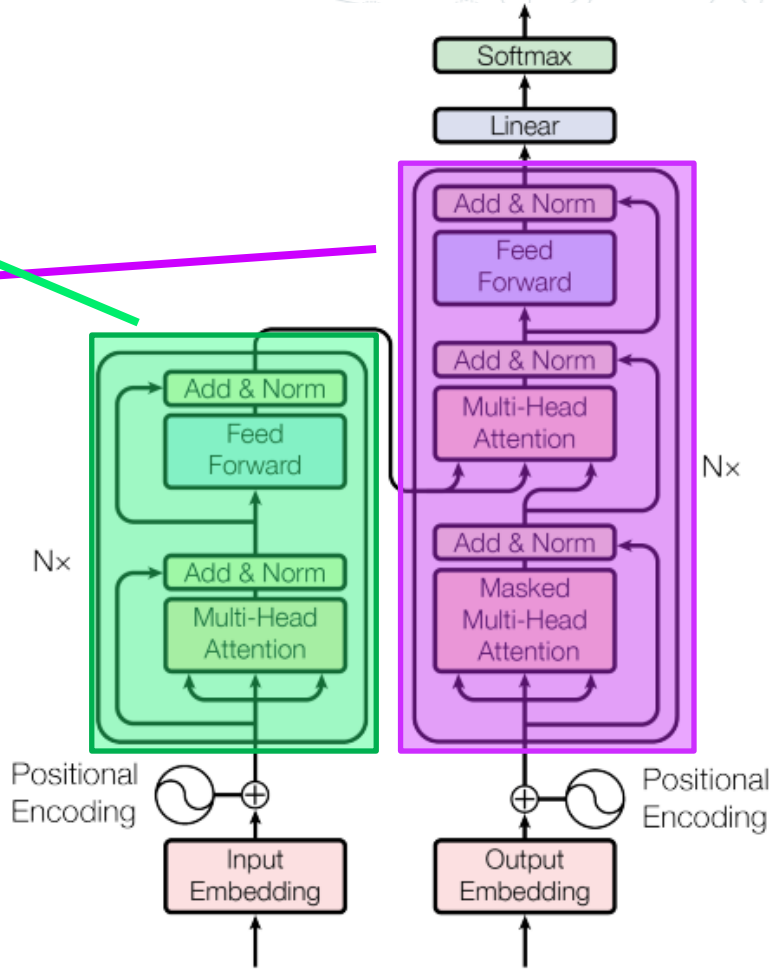
- **Training**
  - **Time:** Around 3 hours to train (6 Epochs)
  - **Heads:** 8
  - **Layers:** 6
- **Inference**
  - **BLEU Score:** 36.0, from 41.8

Model	BLEU	
	EN-DE	EN-FR
ByteNet [18]	23.75	
Deep-Att + PosUnk [39]		39.2
GNMT + RL [38]	24.6	39.92
ConvS2S [9]	25.16	40.46
MoE [32]	26.03	40.56
Deep-Att + PosUnk Ensemble [39]		40.4
GNMT + RL Ensemble [38]	26.30	41.16
ConvS2S Ensemble [9]	26.36	**41.29**
Transformer (base model)	27.3	38.1
Transformer (big)	**28.4**	**41.8**

Portuguese to English - Model

- **Tensorflow Implementation**
  - Custom layers
  - Found dataset **(ted_hrlr_translate)**
  - Found tokenizer **(pt & en variants)**
- **Batch-able on ROSIE**
  - .sh & .py scripts developed, along with Jupyter

```python
class Transformer(tf.keras.Model):
    def __init__(self, *, num_layers, d_model, num_heads, dff,
                 input_vocab_size, target_vocab_size, dropout_rate=0.1):
        super().__init__()
        self.encoder = Encoder(num_layers=num_layers, d_model=d_model,
                               num_heads=num_heads, dff=dff,
                               vocab_size=input_vocab_size,
                               dropout_rate=dropout_rate)

        self.decoder = Decoder(num_layers=num_layers, d_model=d_model,
                               num_heads=num_heads, dff=dff,
                               vocab_size=target_vocab_size,
                               dropout_rate=dropout_rate)

        self.final_layer = tf.keras.layers.Dense(target_vocab_size)

    def call(self, inputs):
        # To use a Keras model with `.fit` you must pass all your inputs in the
        # first argument.
        context, x  = inputs

        context = self.encoder(context)  # (batch_size, context_len, d_model)

        x = self.decoder(x, context)  # (batch_size, target_len, d_model)

        # Final linear layer output.
        logits = self.final_layer(x)  # (batch_size, target_len, target_vocab_size)

        try:
            # Drop the keras mask, so it doesn't scale the losses/metrics.
            # b/250038731
            del logits._keras_mask
        except AttributeError:
            pass

        # Return the final output and the attention weights.
        return logits
```

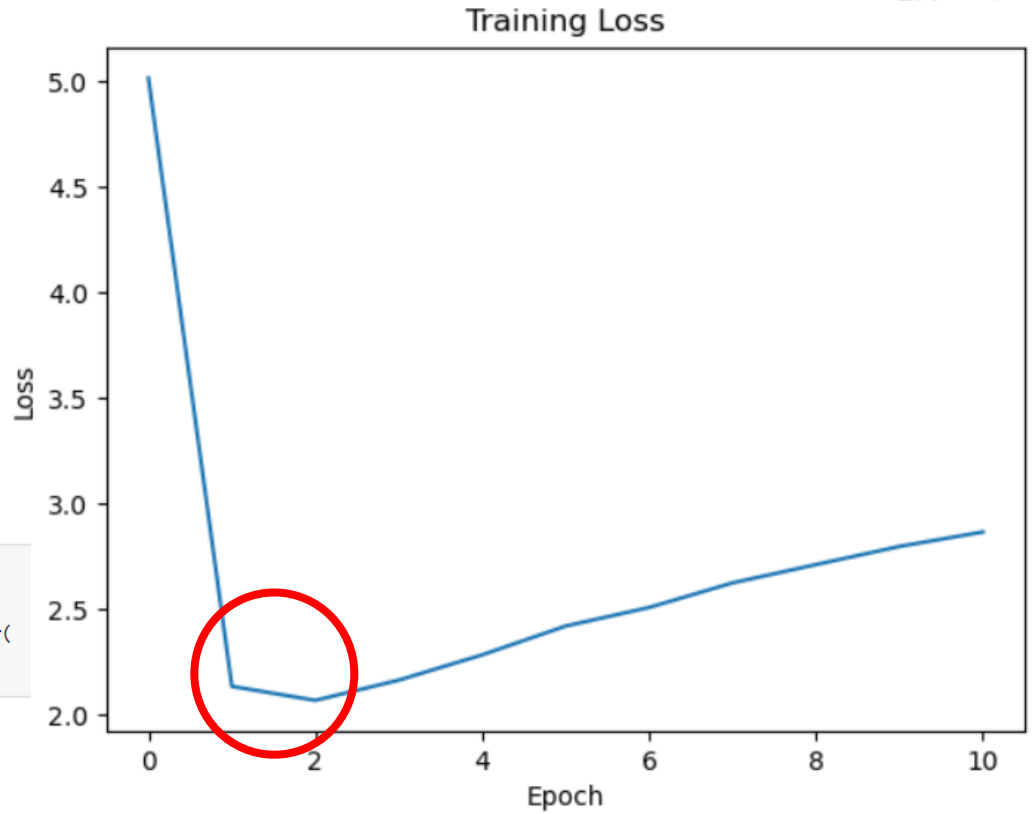


Portuguese to English - Training

Hyperparameters

Attention Heads: 8
Layers: 6
Epochs: 100
Time: 1d 4h 30m

Msk Acc: 71.4%




```
sentence = 'este é um problema que temos que resolver.'
ground_truth = 'this is a problem we have to solve .'

translated_text, translated_tokens, attention_weights = translator(
    tf.constant(sentence))
print_translation(sentence, translated_text, ground_truth)
```

```
Input:         : este é um problema que temos que resolver.
Prediction     : this is a problem that we have to solve .
Ground truth   : this is a problem we have to solve .
```

Discussion

- **What went well and what went poorly. What would you do differently?**
  - **Good:** Lots of resources
  - **Bad:** Lots of resources
    - We'll share our recommended resources soon

What Insights Did We Gain?

- “Attention Is All You Need” Paper
  - Attention
  - Transformers
- Natural Language Processing
- Seq2Seq Models (Encoder-Decoder Usage)
- Tensorflow custom layers

*The **Transformer** has been influential in the space of NLP due to its **deep** and **context-aware** architecture*

Skip Connections

Self-Attention

Allowing for some of the most accurate translation (and other) models

Thank you!

Any Questions?

If you care, here are our recommended learning resources...