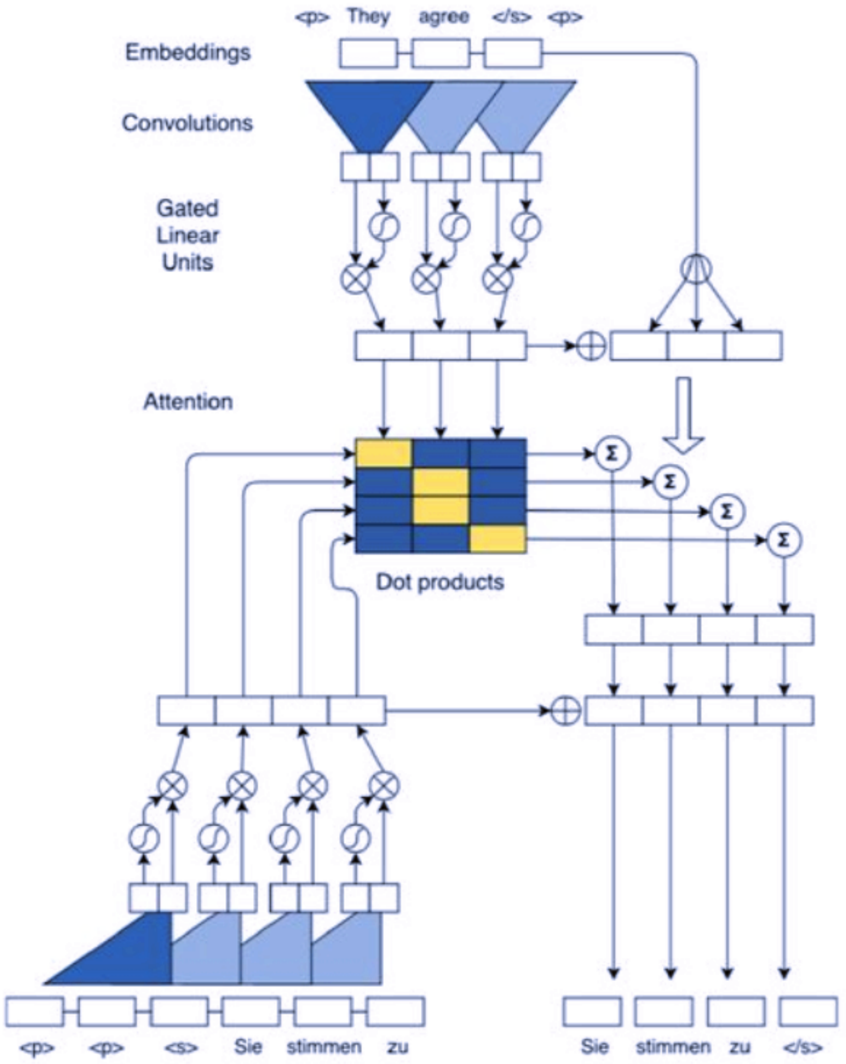
Transformer

Attention Is All You Need

박성진

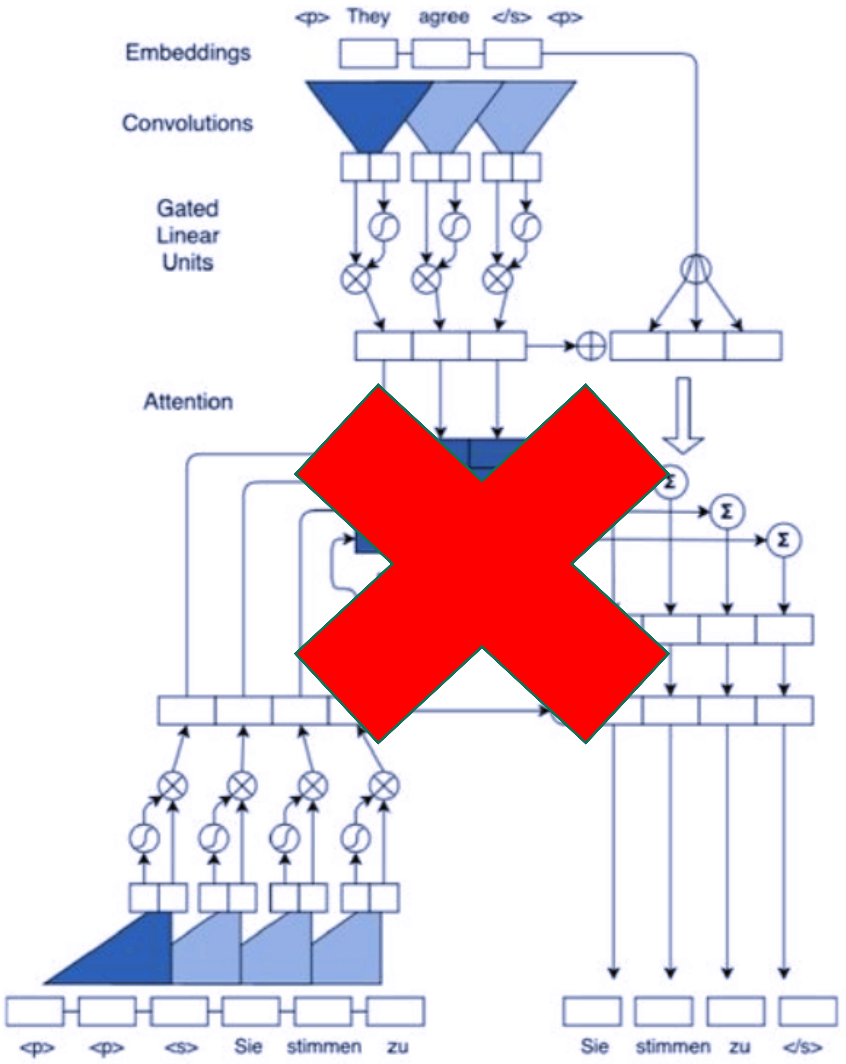
1. Introduction

- Existing NMT architectures
  - RNN + Attention
  - CNN + Attention



1. Introduction

- Existing NMT architectures
  - RNN + Attention
  - CNN + Attention
- This work : Transformer
  - No RNN, No CNN, Only Attention



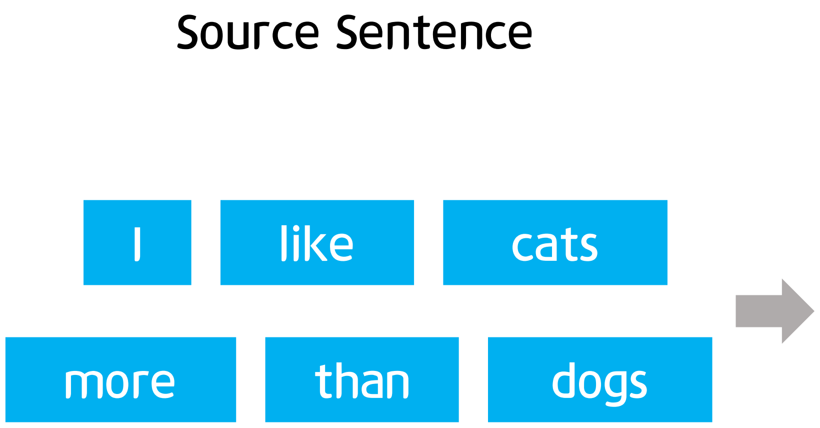
1. Introduction

- Background on NMT

  - Sentences are comprised of words, so this is equivalent to mapping a sequence to another sequence.
  - So people have developed many methods for performing such a mapping:
    these methods are referred to as Sequence-to-Sequence models
  - Sequence-to-Sequence tasks are performed using an Encoder-Decoder model

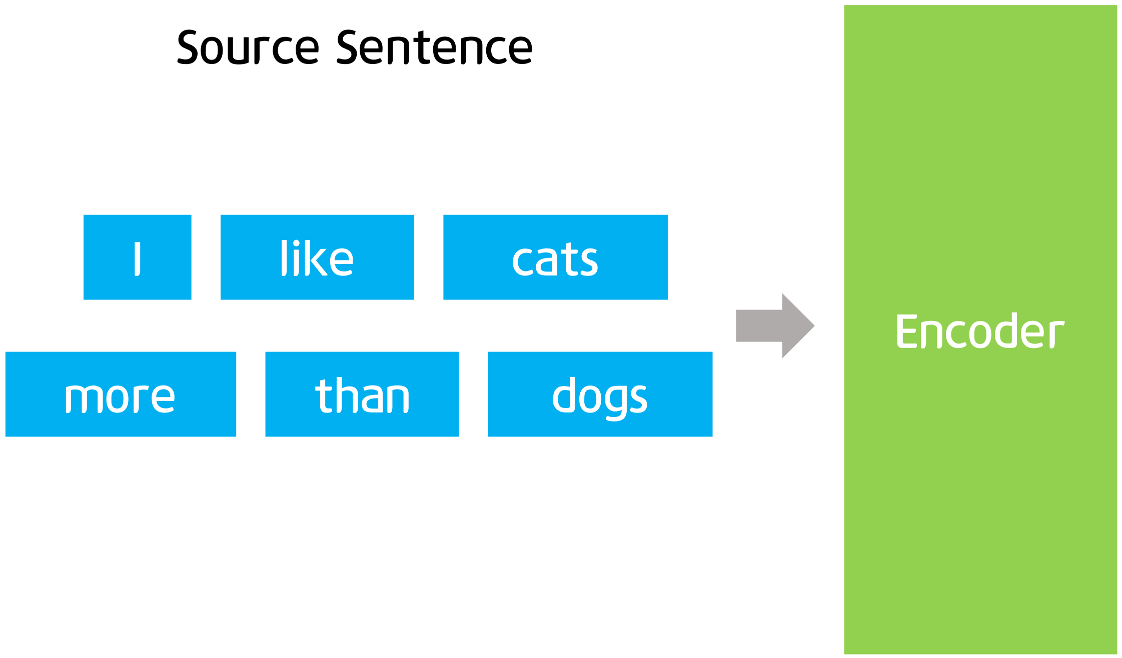
1. Introduction

- Background on NMT



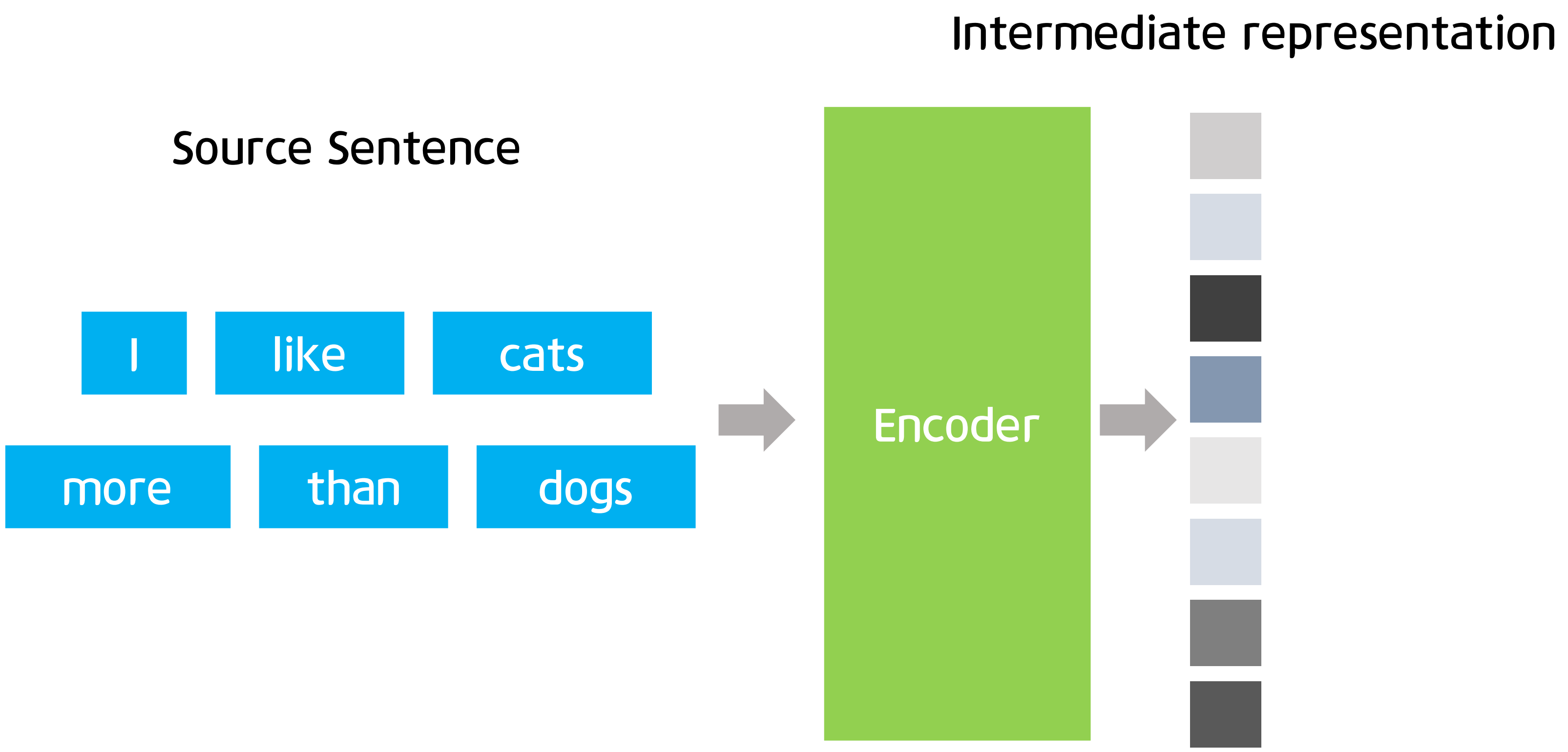
1. Introduction

- Background on NMT



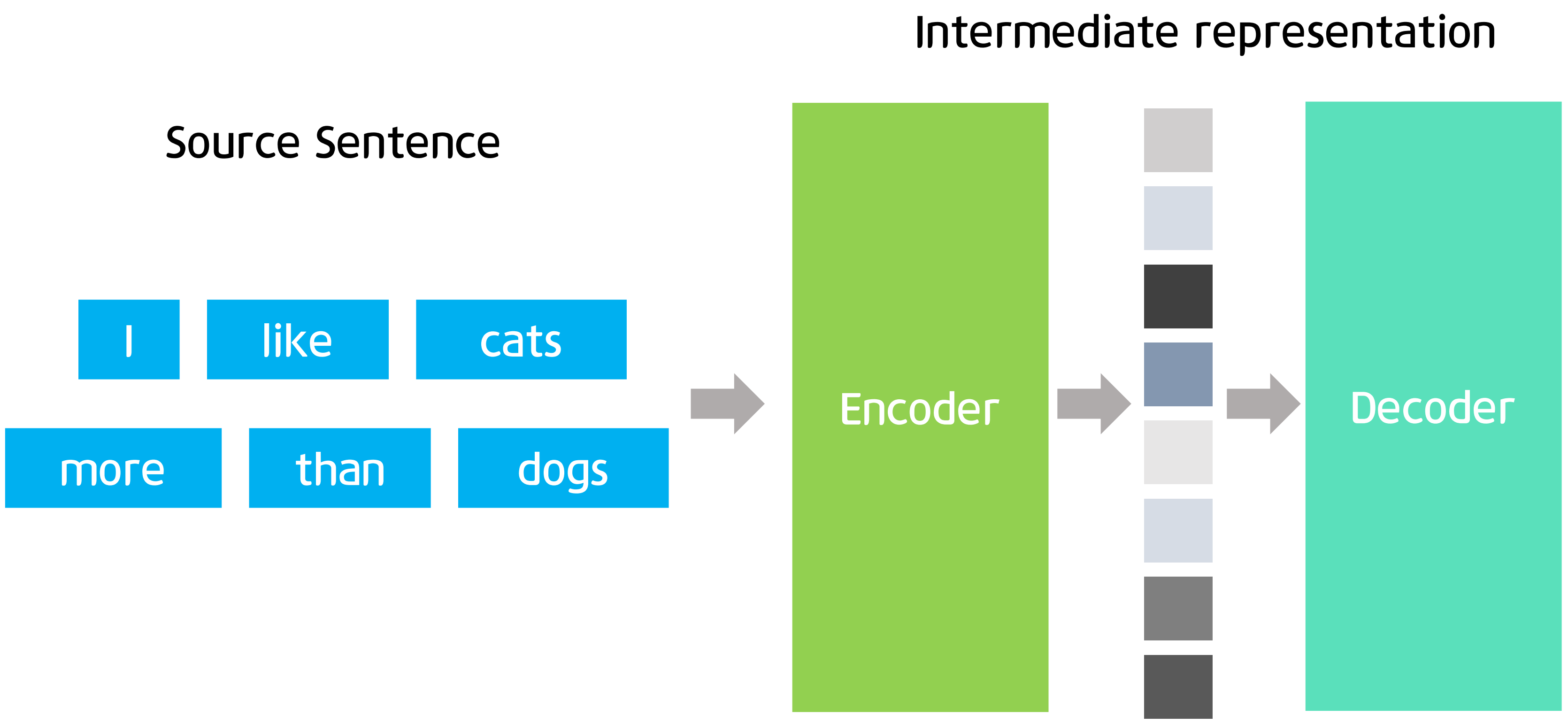
1. Introduction

- Background on NMT



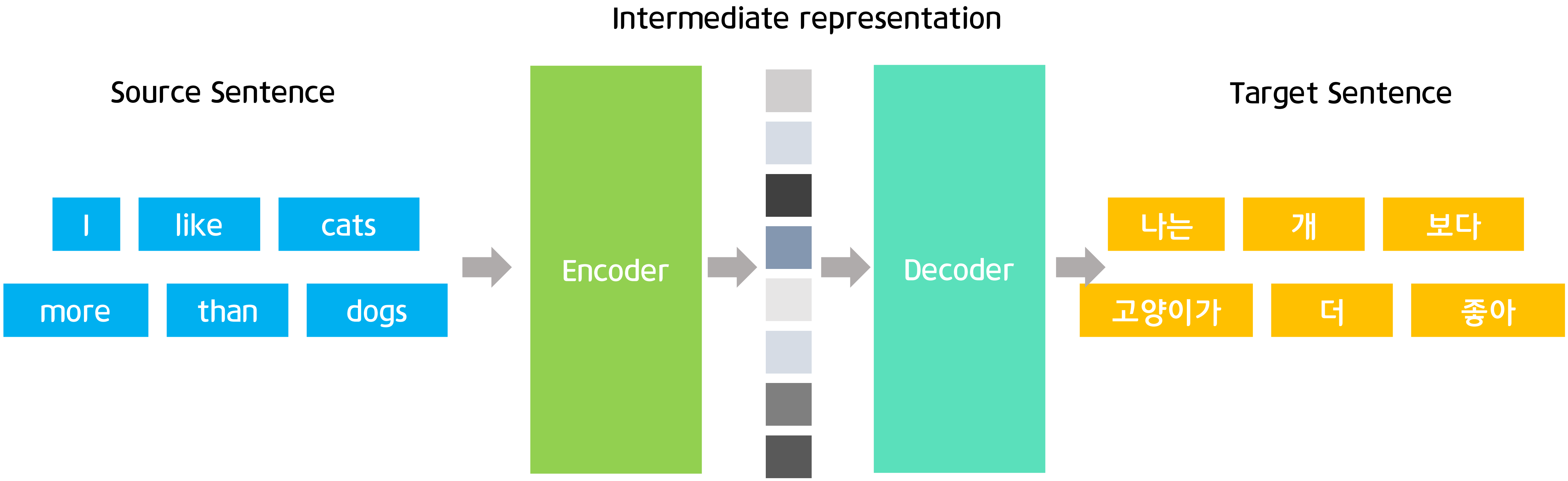
1. Introduction

- Background on NMT



1. Introduction

- Background on NMT



1. Introduction

- Background on NMT
  - The basic idea is
    - Encoder takes the sequence of input words
    - Converts it to some intermediate representation
    - Then passes that representation to the Decoder which produces the target sequence
  - These models are trained to maximize the likelihood of generating the correct output sequence at each step
  - Before Transformer, RNNs were the most widely-used and successful architecture for both Encoder and Decoder.

1. Introduction

- RNNs(LSTM, GRU)
  - RNNs seemed to be born for this task: their recurrent nature perfectly matched the sequential nature of language.
  - Now, this is all great when the sentences are short, but when they become longer we encounter a problem.

1. Introduction

- RNNs

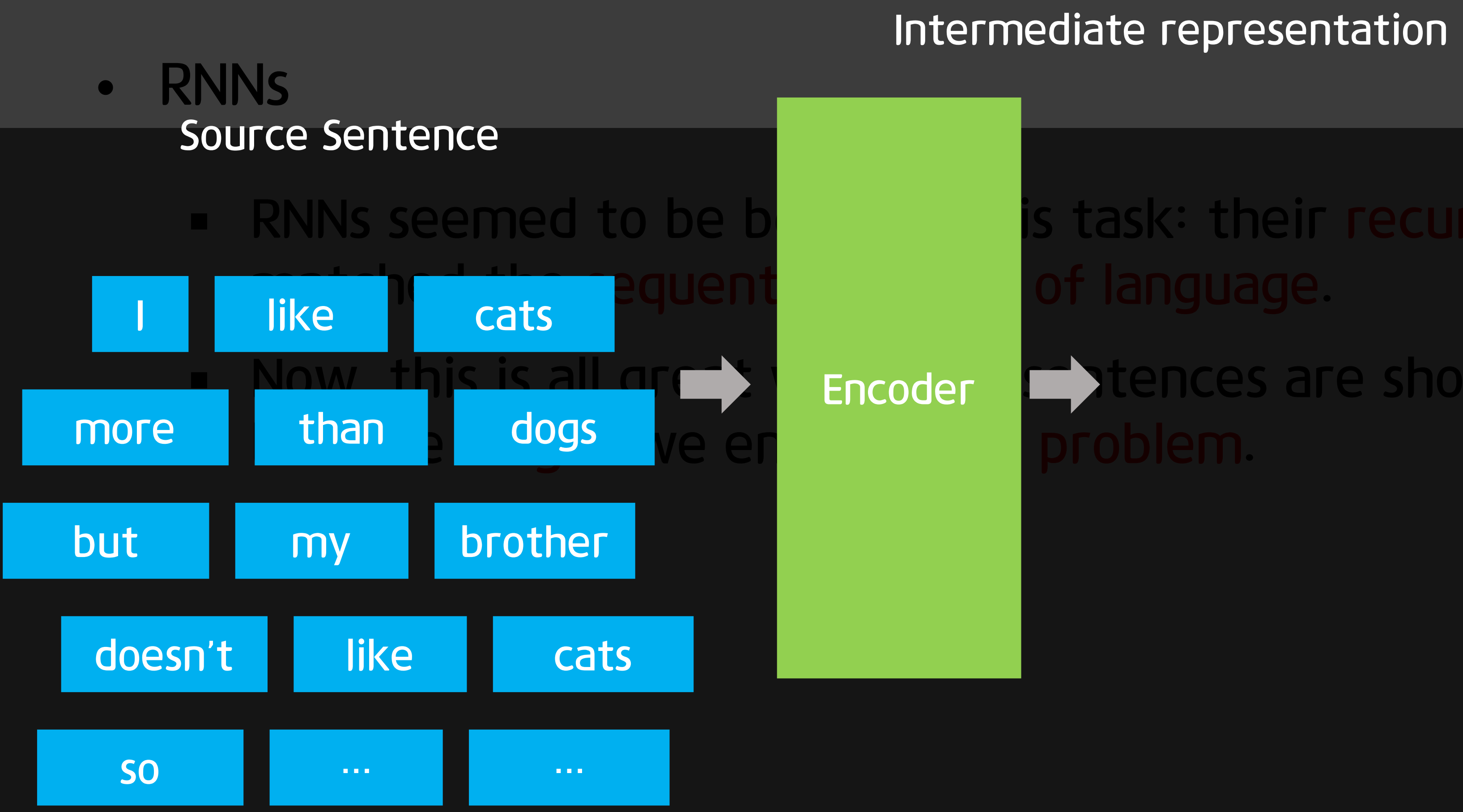
Source Sentence

  - RNNs seemed to be born for this task: their recurrent nature perfectly matched the sequential nature of language.
  - Now, this is all great when the sentences are short, but when they ... we encounter a problem.

I like cats
more than dogs
but my brother
doesn't like cats
so … …

1. Introduction

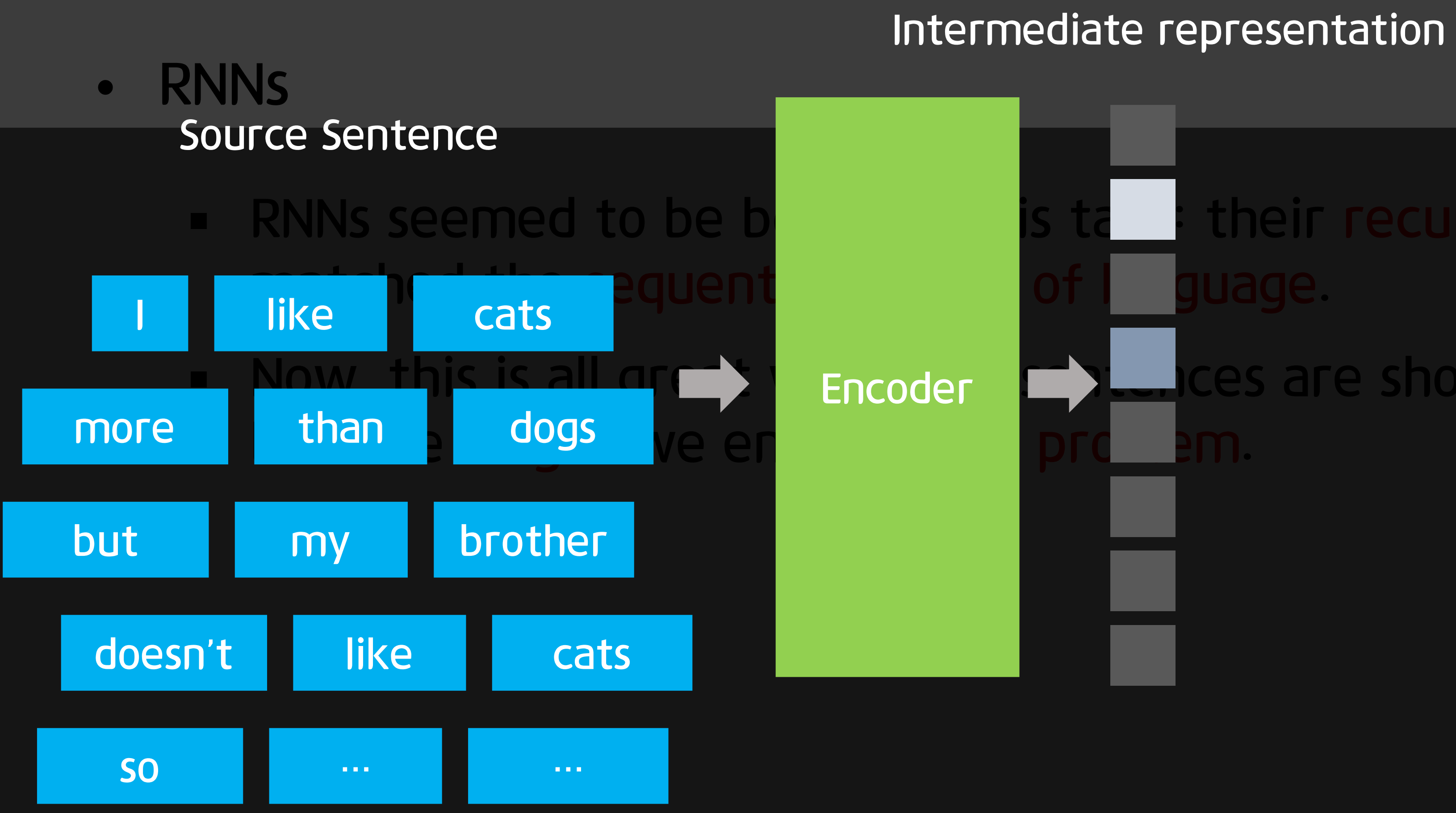
- RNNs



  - RNNs seemed to be b… is task: their recurrent nature perfectly matched the sequent… of language.
  - Now, this is all gr… tences are short, but when they … we en… problem.

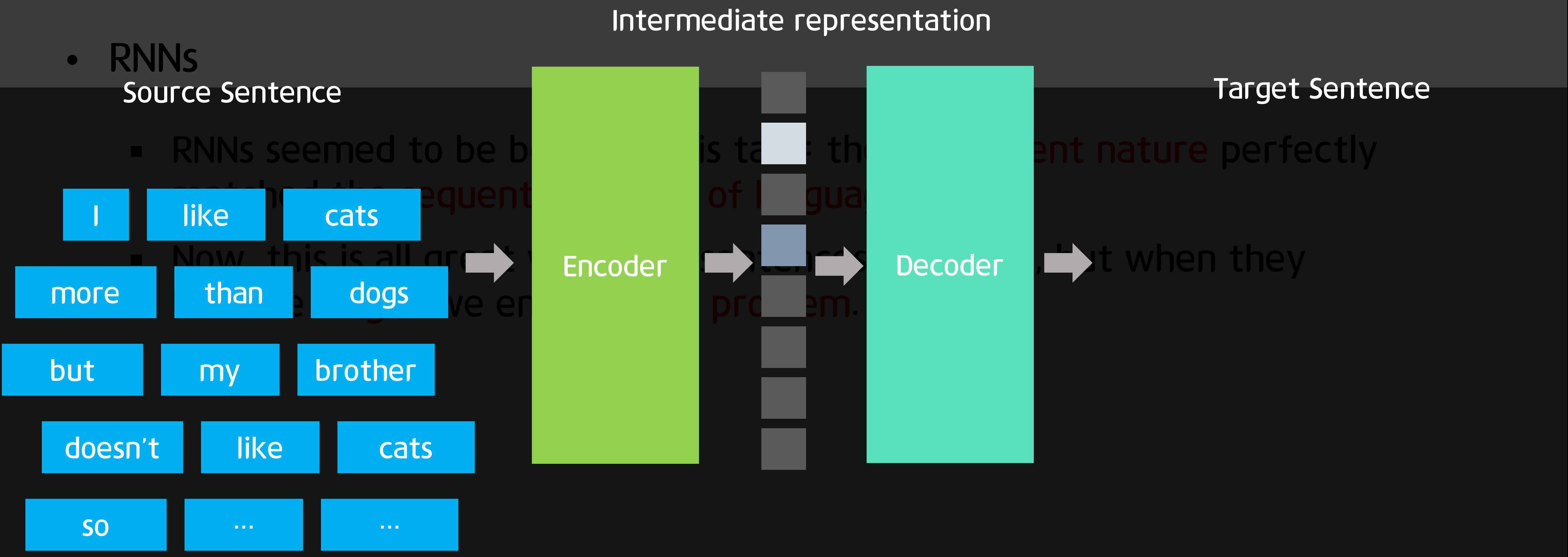
1. Introduction

- RNNs
  - RNNs seemed to be b… is ta… their recurrent nature perfectly … equent… of l… guage.
  - Now, this is all gr… … nces are short, but when they … we e… pro… em.



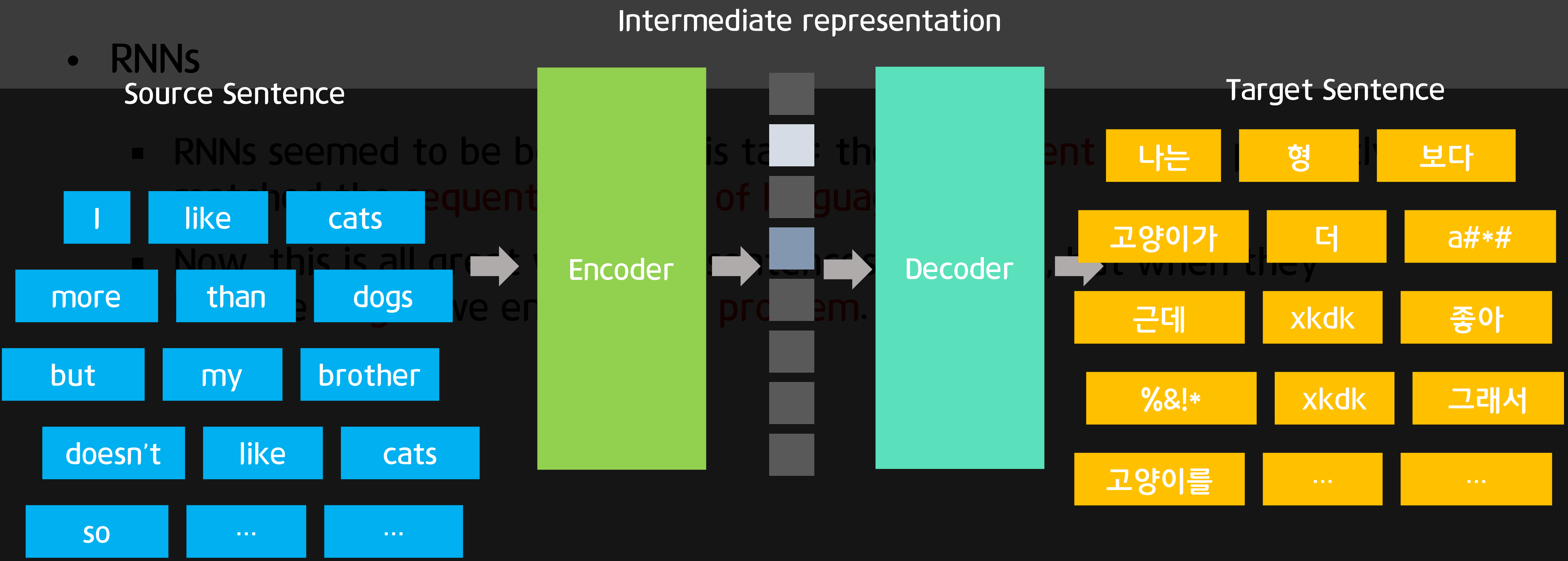
1. Introduction

- RNNs



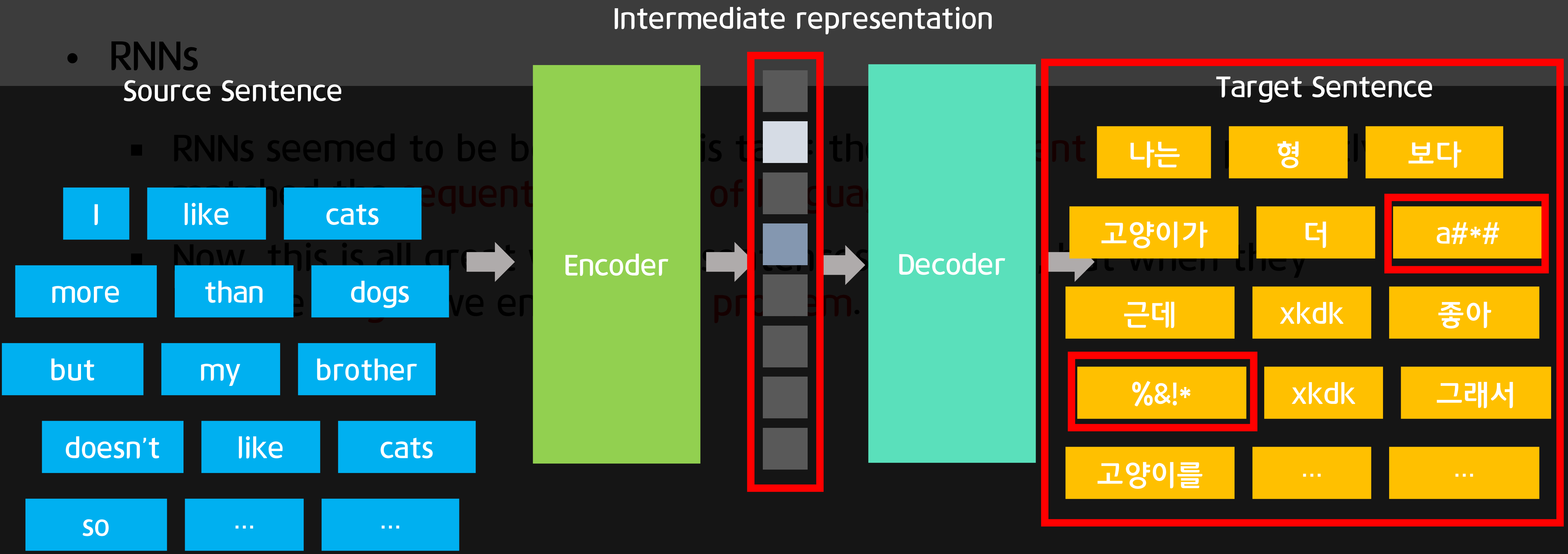
1. Introduction

- RNNs



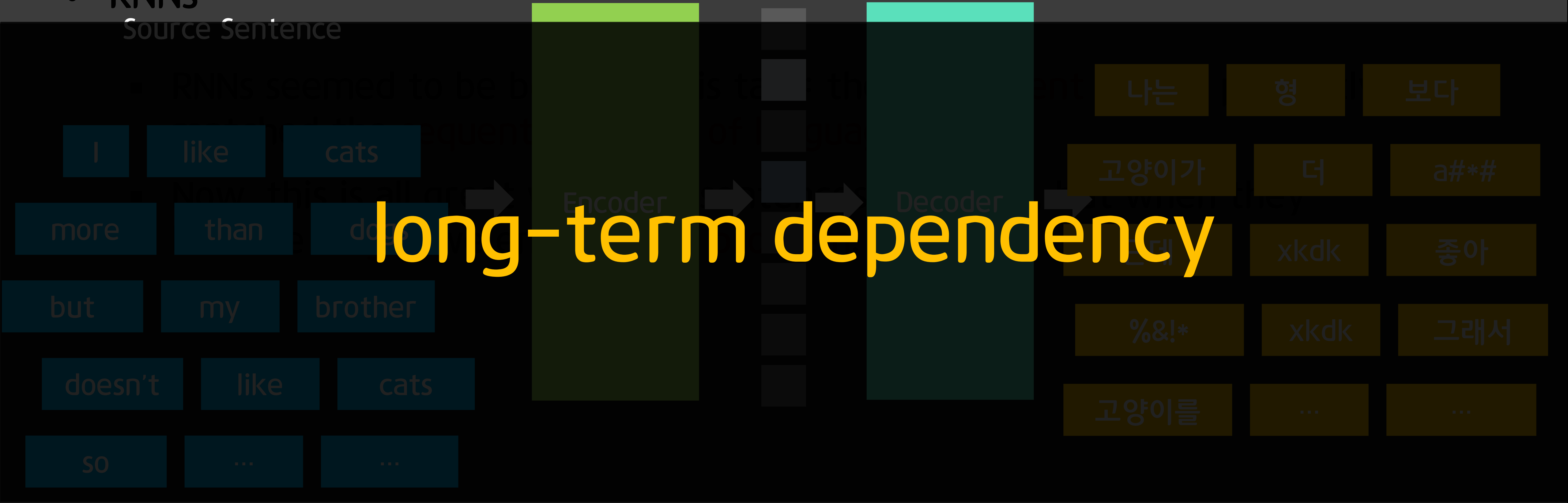
1. Introduction

- RNNs



1. Introduction

- RNNs

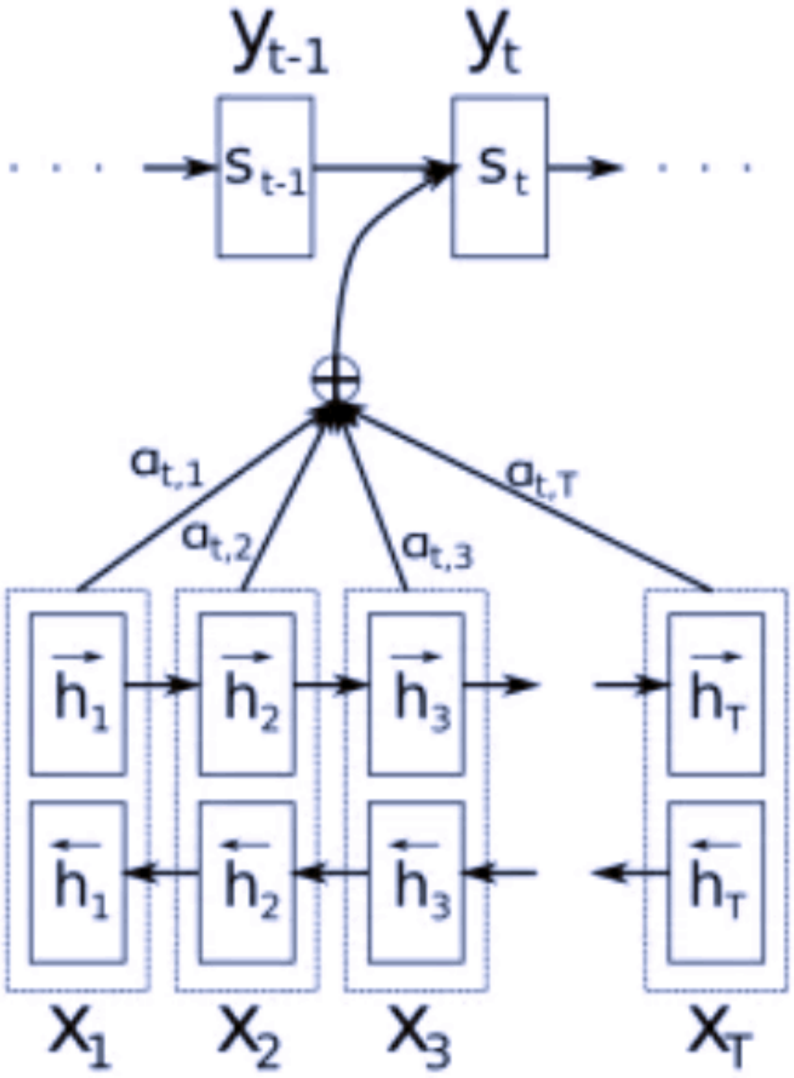


1. Introduction

- Attention mechanism
  - Intuitively, the attention mechanism allows the decoder to "look back" at the entire sentence and selectively extract the information it needs during decoding.
  - Concretely, attention gives the decoder access to all the encoder's hidden states.
  - So what attention does is it asks the decoder to choose which hidden states to use and which to ignore by weighting the hidden states.
  - The decoder is then passed a weighted sum of hidden states to use to predict the next word.

1. Introduction

- Attention mechanism
  - The decoder state is used to compute the attention weights of the hidden encoder states.
  - The attention weights change for each decoder state, and the model learns to "focus" on the relevant parts of the input.



1. Introduction

- A Few shortcomings of RNNs
  - The sequential nature of RNNs. When we process a sequence using RNNs, each hidden state depends on the previous hidden state. This becomes a major pain point on GPUs
  - The other is the difficulty of learning long-range dependencies in the network

1. Introduction

- A Few shortcomings of RNNs
  - The sequential nature of RNNs. When we process a sequence using RNNs, each hidden state depends on the previous hidden state. This becomes a major pain point on GPU
  - The [illegible] long-range dependencies in the network

Didn't LSTM handle the long-range dependency problem in RNNs?

1. Introduction

- A Few shortcomings of RNNs
  - The sequential nature of RNNs. When we process a sequence using RNNs, each hidden state depends on the previous hidden state. This becomes a major pain point on GPUs.
  - The other is the difficulty of learning long-range dependencies in the network

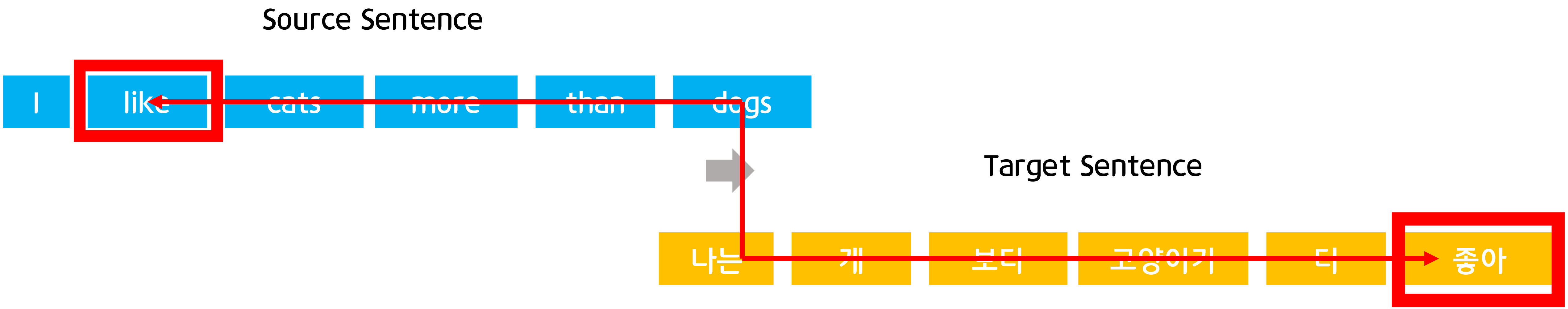
Didn't we introduce Attention to handle this problem?

1. Introduction

- A Few shortcomings of RNNs
  - But remembering things for long periods is still a challenge, and RNNs can still have short-term memory problems
  - Furthermore, some words have multiple meanings that only become apparent in context

1. Introduction

- A Few shortcomings of RNNs

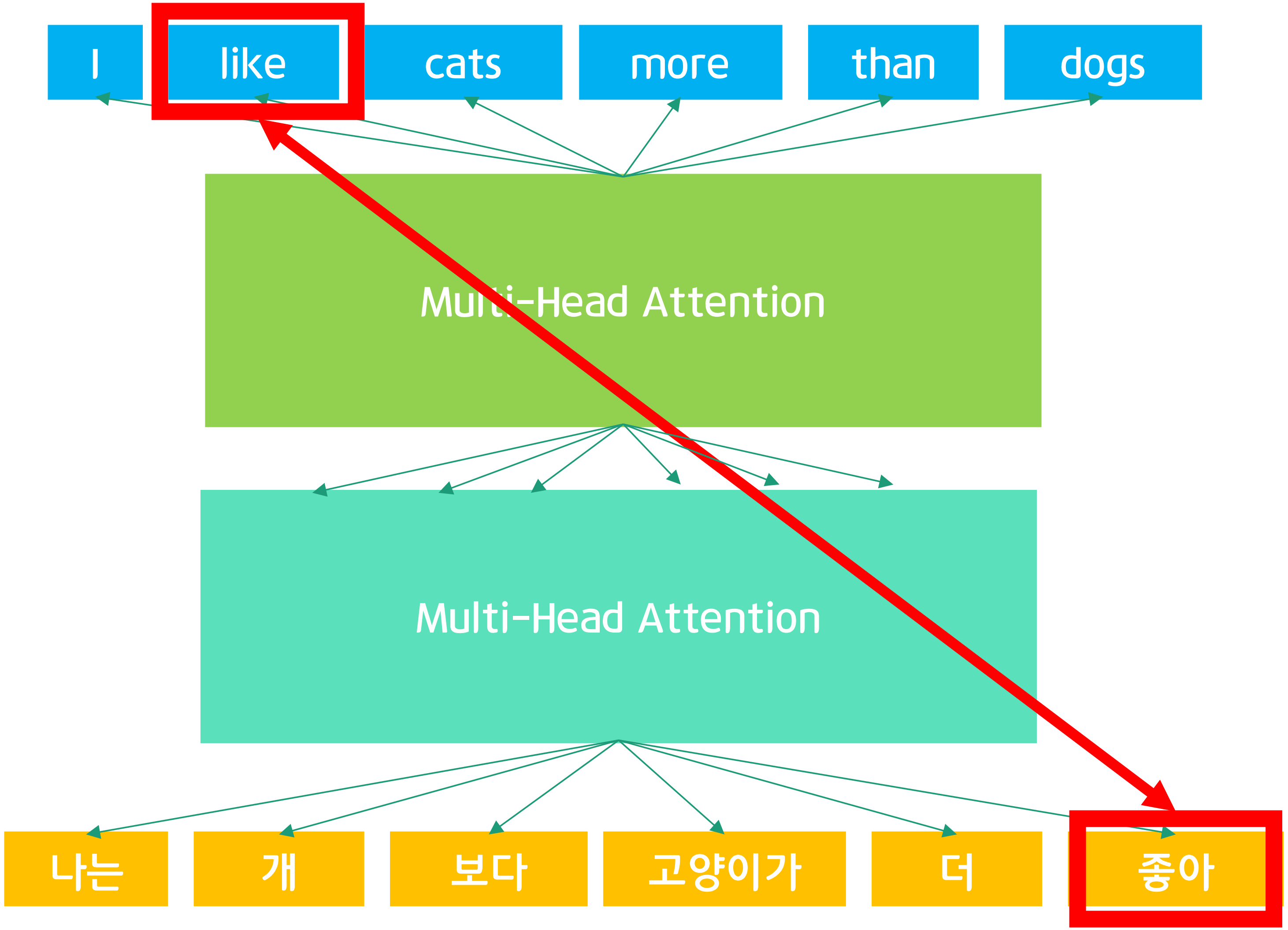


2. Multi head Attention

- Multi head Attention
  - Why don't we just allow the encoder and decoder to see the entire input sequence all at once, directly modeling these dependencies using attention?
  - This is the basic idea behind the Transformer

2. Multi head Attention

- Multi head Attention

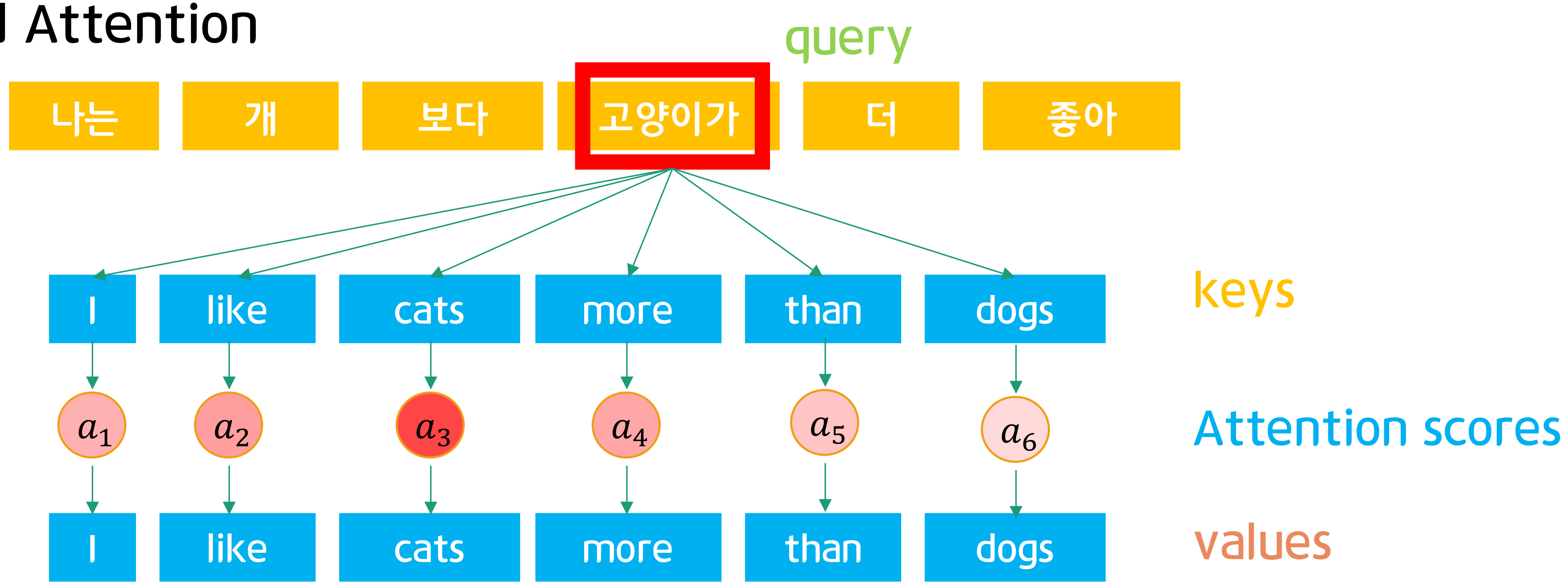


2. Multi head Attention

- Multi head Attention
  - The attention mechanism in the Transformer is interpreted as a way of computing the relevance of a set of values(information) based on some keys and queries.
  - Traditionally, the attention weights were the relevance of the encoder hidden states(values) in processing the decoder state(query)
  - And were calculated based on the encoder hidden states(keys) and the decoder hidden state(query)

2. Multi head Attention

- Multi head Attention



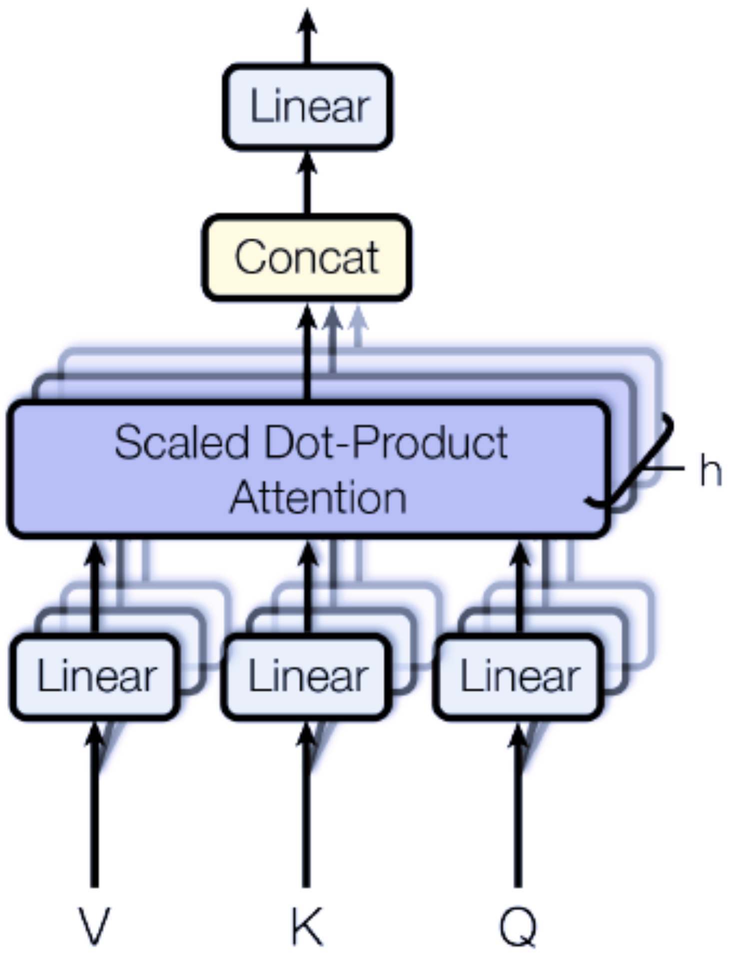
  - The query is the word being decoded ("고양이가" which means cats) and both the keys and values are the source sentence.
  - The attention score represents the relevance, and in this case is large for the word "cats" and small for others.

2. Multi head Attention

- Multi head Attention
  - If we only computed a single attention weighted sum of the values, it would be difficult to capture various different aspects of the input.
  - For instance, in the sentence “I like cats more than dogs”, you might want to capture the fact that the sentence compares two entities, while also retaining the actual entities being compared.
  - To solve this problem the Transformer uses the Multi-Head Attention block. This block computes multiple attention weighted sums instead of a single attention pass over the values.

2. Multi head Attention

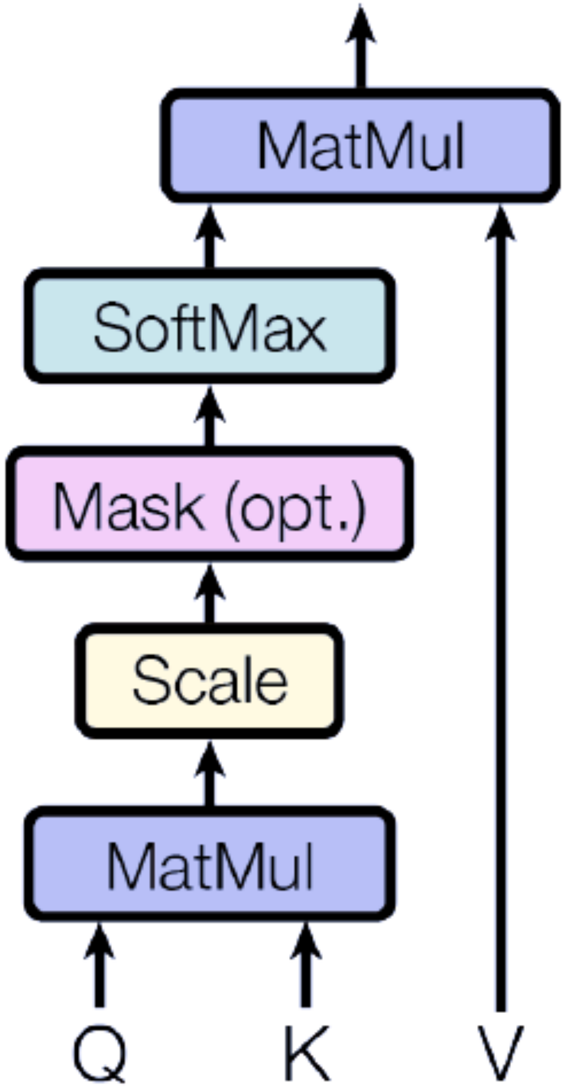
- Multi head Attention
    - Multi-Head Attention applies different linear transformations to the values, keys, and queries for each “head” of attention to learn diverse representations.



Multi-head Attention

3. Scaled Dot Product Attention

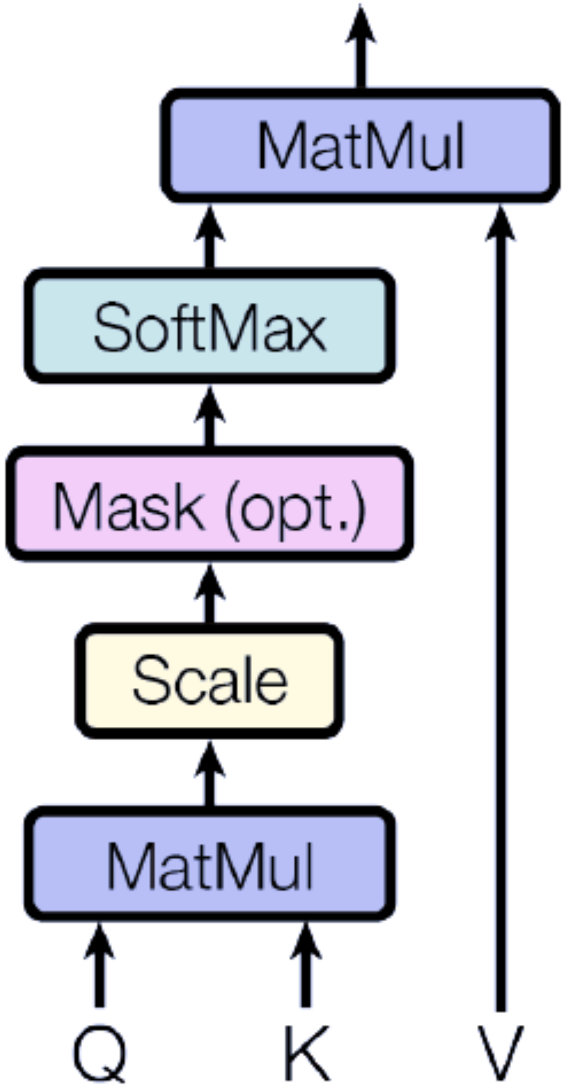
- Scaled Dot Product Attention
  - As for the attention mechansim, the Transformer uses a particular form of attention called the "Scaled Dot-Product Attention"



Scaled Dot Product Attention

3. Scaled Dot Product Attention

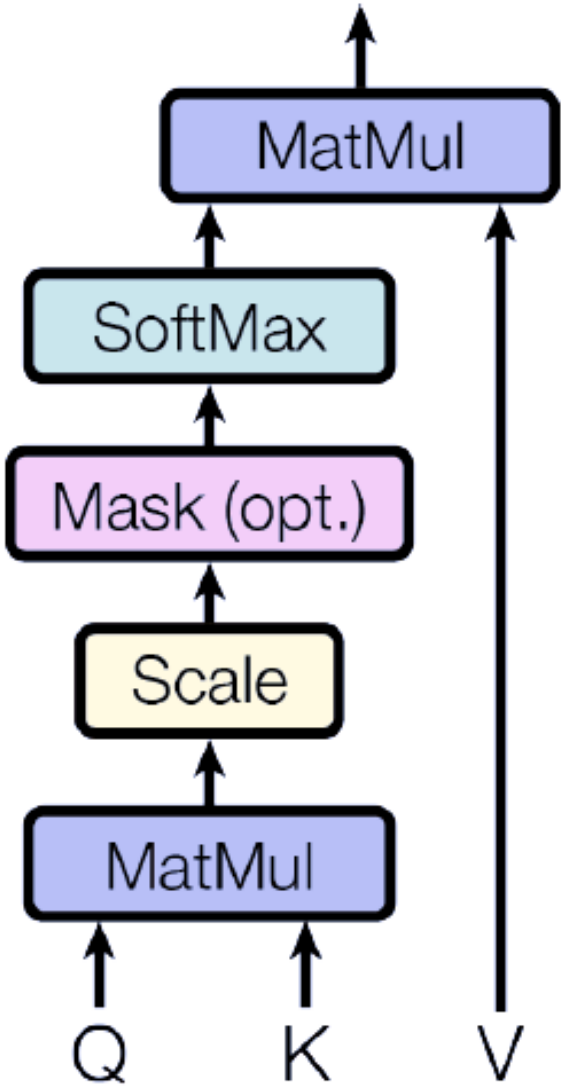
- Scaled Dot Product Attention
  - $\boldsymbol{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{softmax}\left(\frac{\boldsymbol{QK^T}}{\sqrt{\boldsymbol{d_k}}}\right)\mathbf{V}$
  - where Q is the matrix of queries packed together and K, V are the matrices of keys and values packed together.
  - d_k represents the dimensionality of the queries, keys.



Scaled Dot Product Attention

3. Scaled Dot Product Attention

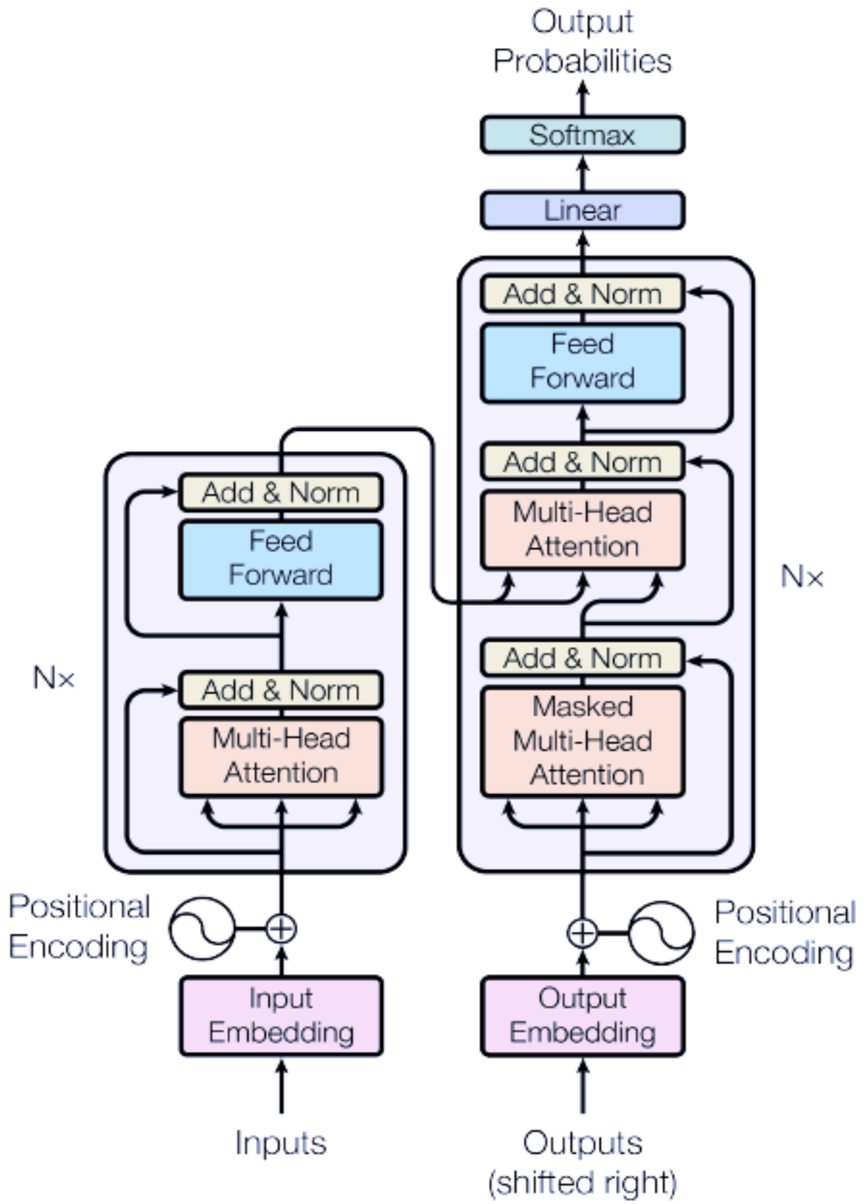
- Scaled Dot Product Attention
  - The basic attention mechanism is simply a dot product between the query and the key.
  - The size of the dot product tends to grow with the dimensionality of the query and key vectors though
  - So the Transformer rescales the dot product to prevent it from exploding into huge values.



Scaled Dot Product Attention

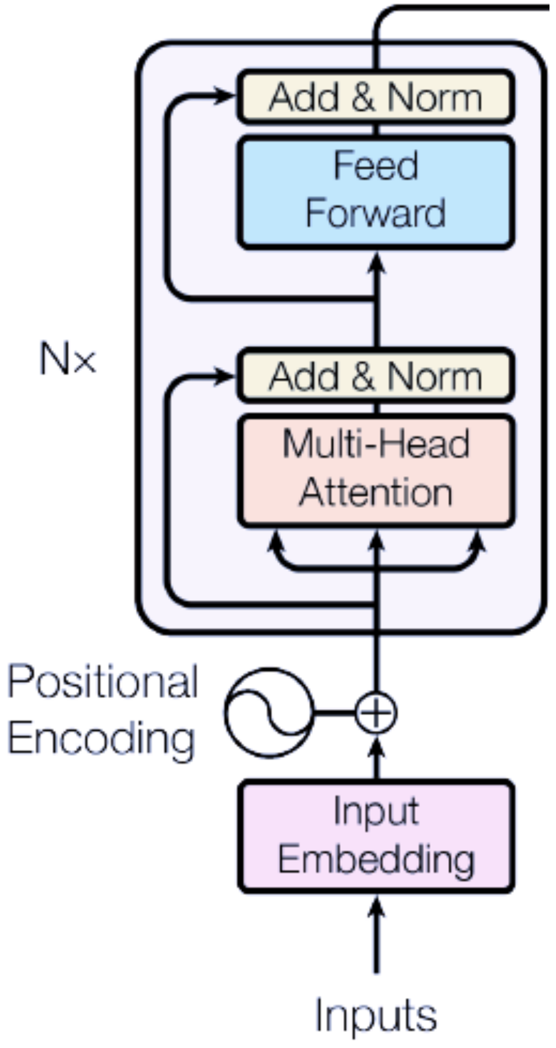
4. The Overall Architecture

- Transformer
  - The basic Encoder-Decoder design of traditional NMT
  - Encoder inputs: embeddings of the input sequence
  - Decoder inputs: the embeddings of the outputs up to that point.
  - 6 blocks for both networks



4. The Overall Architecture

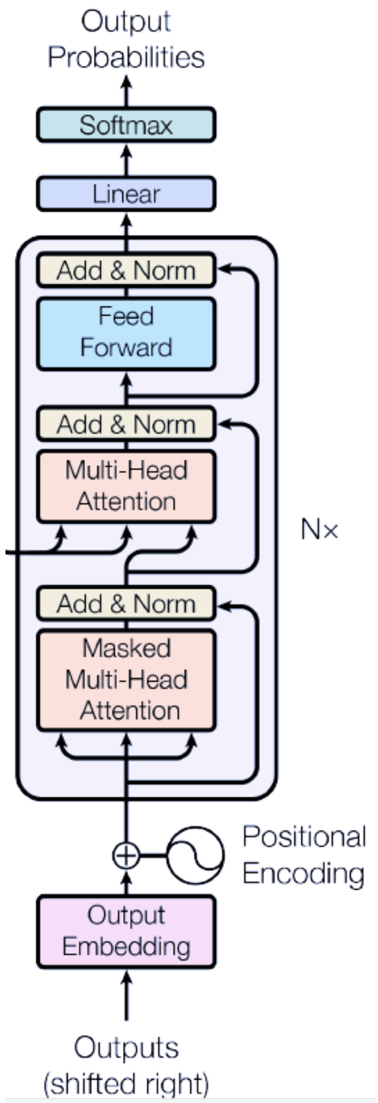
- Transformer - Encoder
  - The encoder is composed of two blocks which are called sub-layers
  - One is the Multi-Head Attention sub-layer over the inputs, mentioned above.
  - The other is a simple feed-forward network.
  - Between each sub-layer, there is a residual connection followed by a layer normalization.
  - $y = LayerNorm(x + Sublayer(x))$



Encoder

4. The Overall Architecture

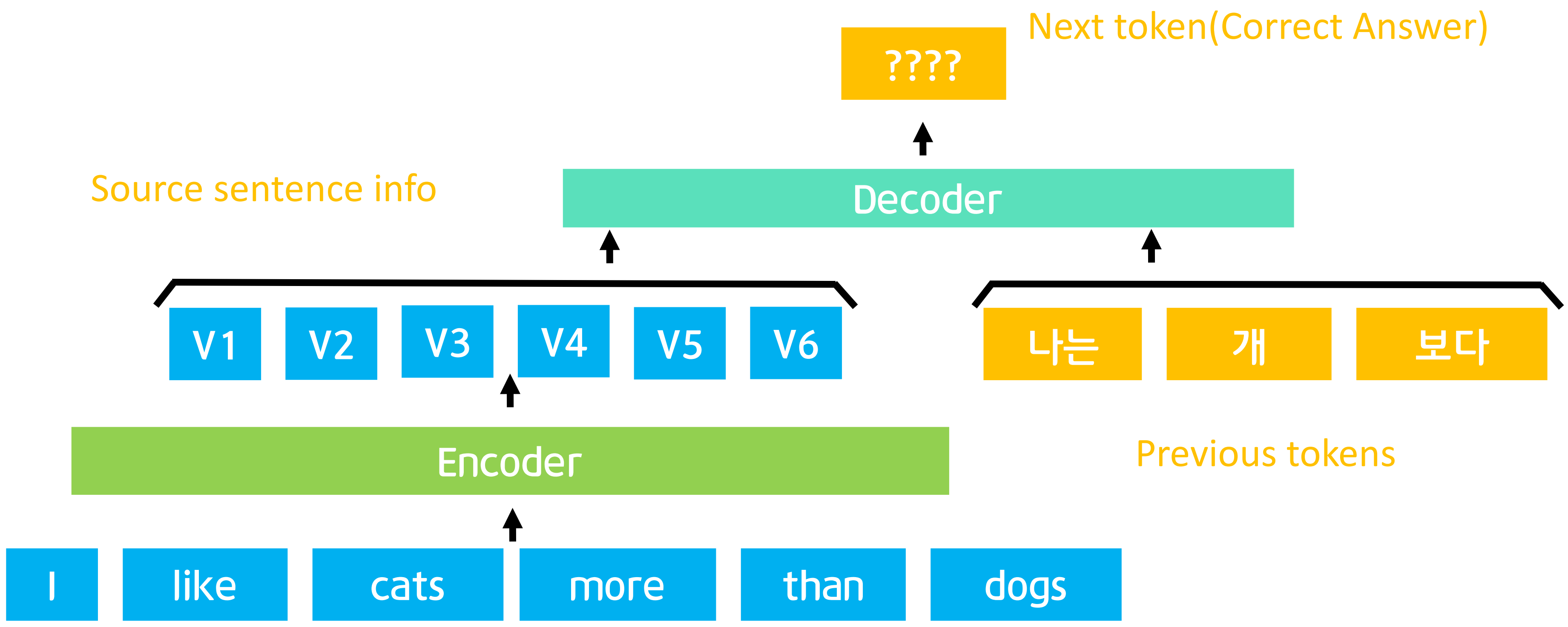
- Transformer – Decoder
  - The decoder is very similar to the encoder
  - But it has one additional Multi-Head Attention layer labeled the "masked multi-head attention" network.
  - This network attends over the previous decoder states, so plays a similar role to the decoder hidden state in traditional machine translation architectures



Decoder

4. The Overall Architecture

- Transformer - Decoder(Masked multi-head attention)



  - We need to mask the inputs to the decoder from future time-steps

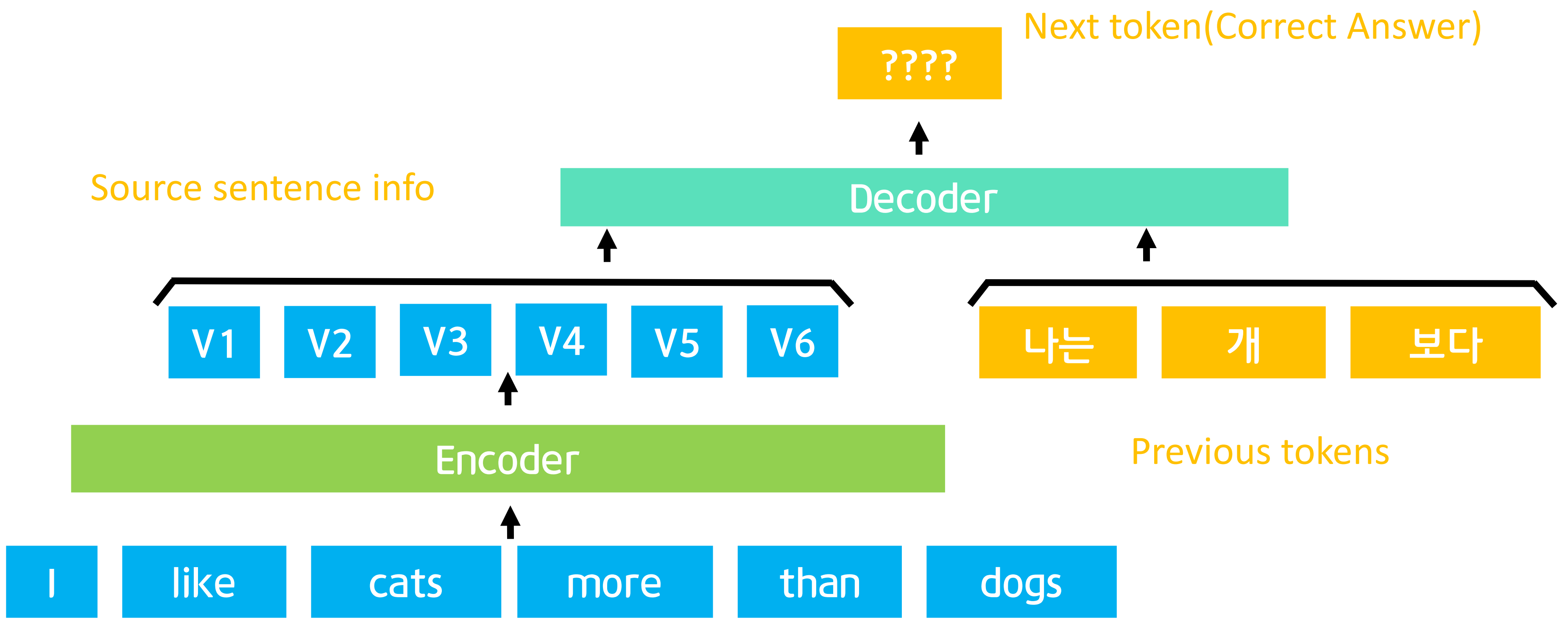
4. The Overall Architecture

- Transformer – Decoder(Masked multi-head attention)

- We need to mask the inputs to the decoder from future time-steps

4. The Overall Architecture

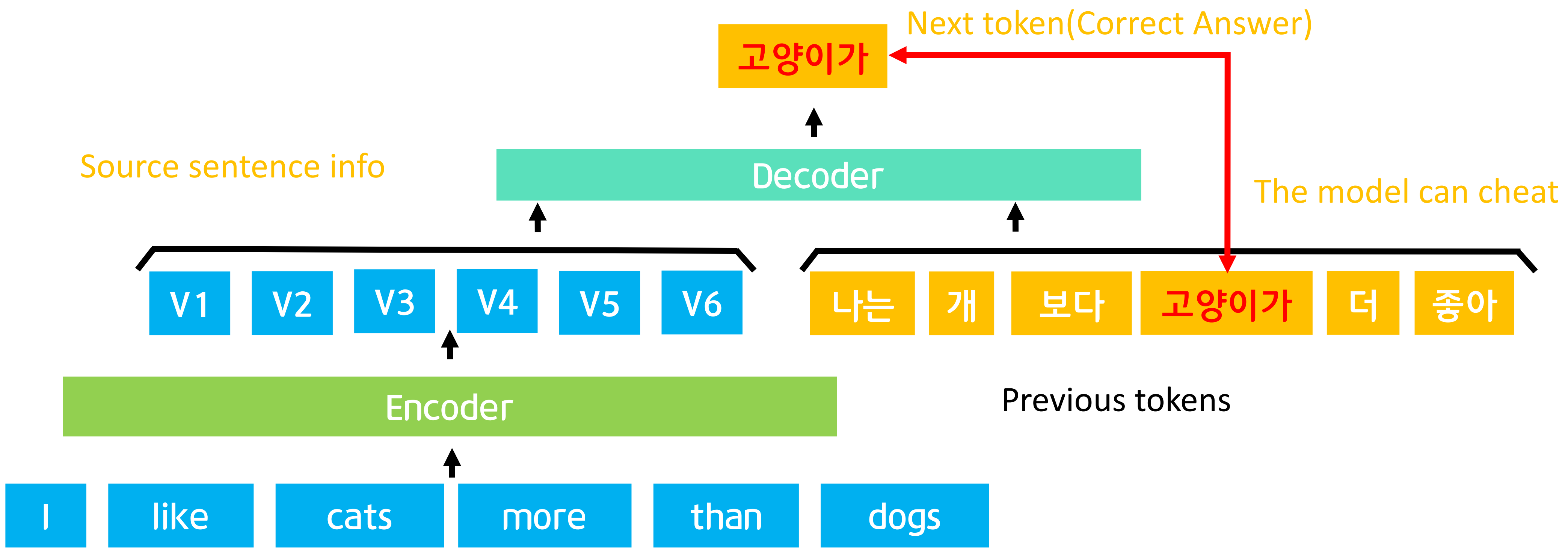
- Transformer - Decoder(Masked multi-head attention)



  - Train “I like cats more than dogs” to “나는 개보다 고양이가 더 좋아”
  - Training the network to predict the word “고양이가” comes after “나는 개보다” when the source sentence is “I like cats more than dogs”

4. The Overall Architecture

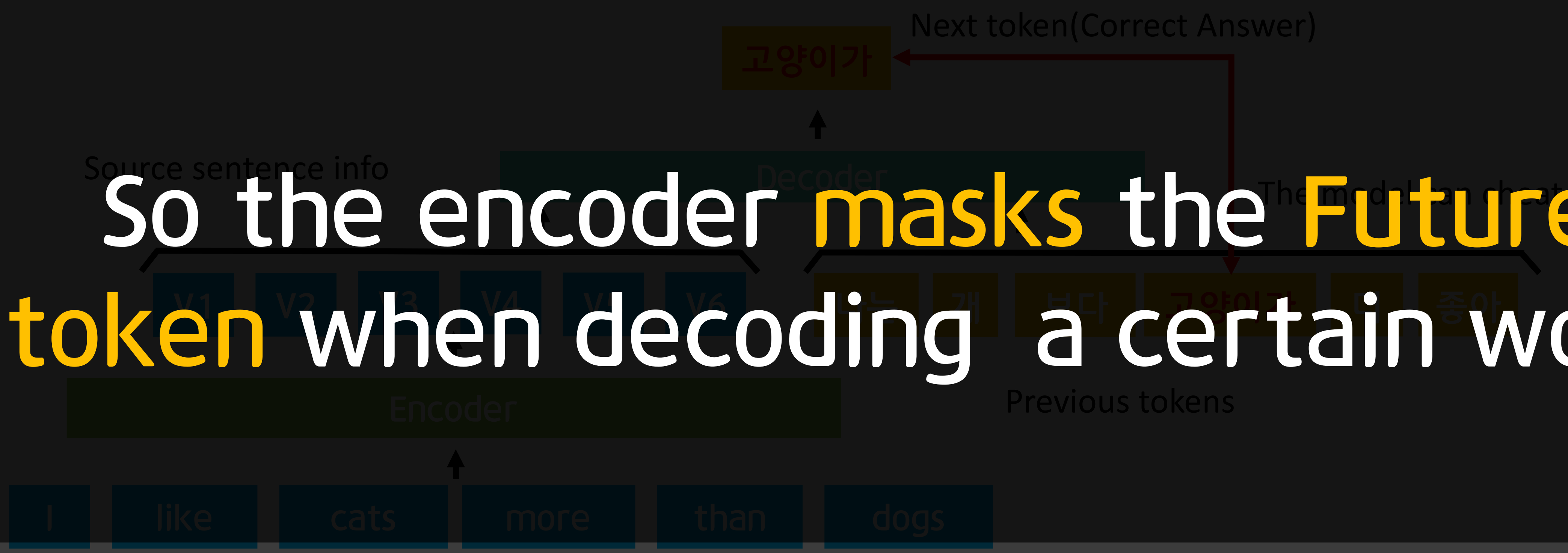
- Transformer - Decoder(Masked multi-head attention)



- In Transformer, we want to process all the sentences at the same time
- If then, The decoder can access to the entire target sentence

4. The Overall Architecture

- Transformer - Decoder(Masked multi-head attention)

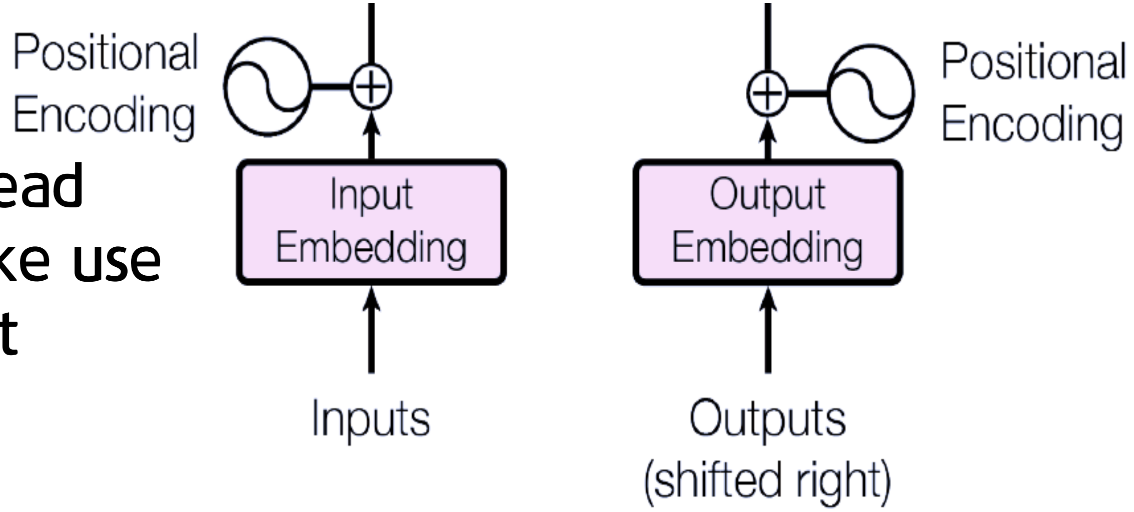


So the encoder masks the Future token when decoding a certain word

- In Transformer, we want to process all the sentences at the same time
- If then, The decoder can access to the entire target sentence

4. The Overall Architecture

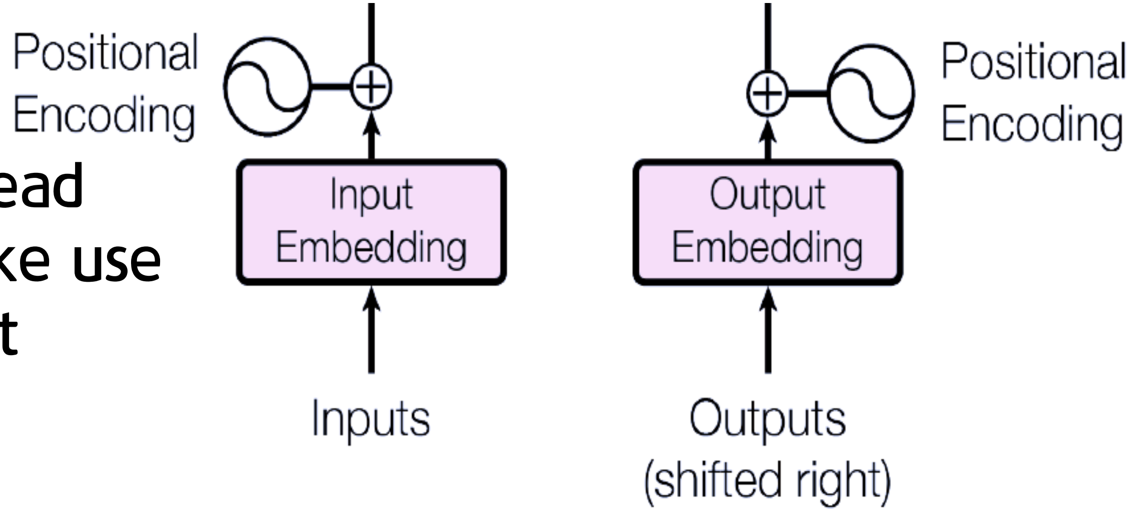
- Transformer – Positional Encodings
  - Unlike recurrent networks, the multi-head attention network cannot naturally make use of the position of the words in the input sequence.



Positional Encoding

4. The Overall Architecture

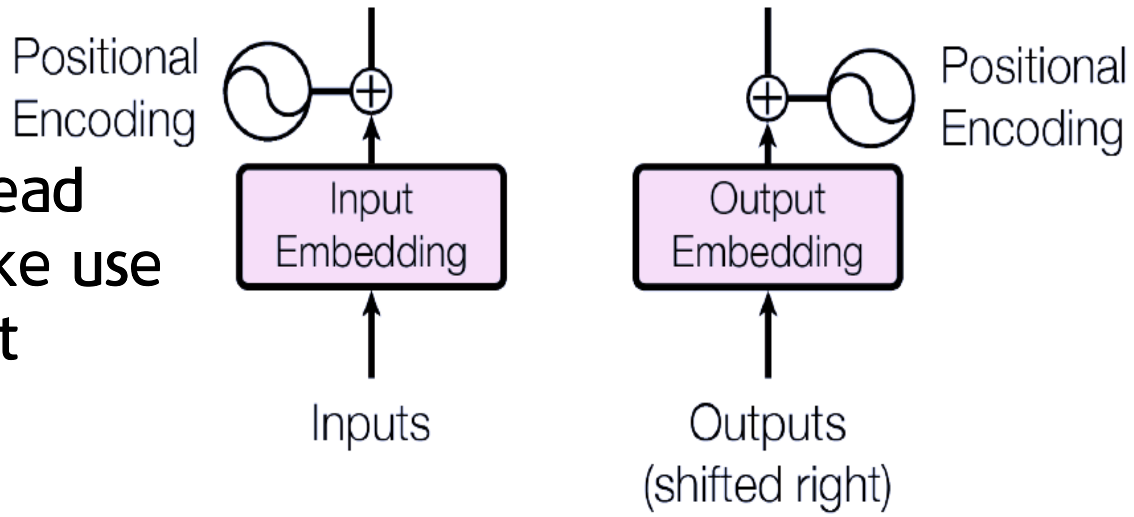
- Transformer – Positional Encodings
  - Unlike recurrent networks, the multi-head attention network cannot naturally make use of the position of the words in the input sequence.
  - positional encodings explicitly encode the relative/absolute positions of the inputs as vectors and are then added to the input embeddings.



Positional Encoding

4. The Overall Architecture

- Transformer – Positional Encodings
  - Unlike recurrent networks, the multi-head attention network cannot naturally make use of the position of the words in the input sequence.
  - positional encodings explicitly encode the relative/absolute positions of the inputs as vectors and are then added to the input embeddings.
  - In this work, we use sine and cosine functions of different frequencies



Positional Encoding

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

5. Training details

- Optimizer
  - Adam Optimizer with $\boldsymbol{B_1 = 0.9, B_2 = 0.98}$
  - Learning rate schedule where they gradually warmed up the learning rate, then decreased it
  - $\boldsymbol{lrate = d_{model}^{-0.5} \cdot min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5})}$

5. Training details

- Regularization
  - Residual Dropout
  - Dropout to each sublayer before adding it to the original input.
  - Dropout to the sum of the embeddings and to the positional encodings. (0.1 by default)
  - Label smoothing
  - To penalize the model when it becomes too confident in its predictions
  - It hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score

6. Results And Discussion

- Results for English-to-German translation and English-to-French translation

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

6. Results And Discussion

- The configuration of the model

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

6. Results And Discussion

- Discussion
  - The first sequence transduction model based entirely on Attention (Multi-head attention)
  - Faster than architectures based on recurrent or convolutional layers
  - Achieved a new state of the art(2014)

6. References

- https://mchromiak.github.io/articles/2017/Sep/12/Transformer-Attention-is-all-you-need/#.XIt_oxMzb0R

- https://slideplayer.com/slide/13789541/

- https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

- http://mlexplained.com/2017/12/29/attention-is-all-you-need-explained/

감사합니다