注意力，Transformer，BERT，GPT，BART

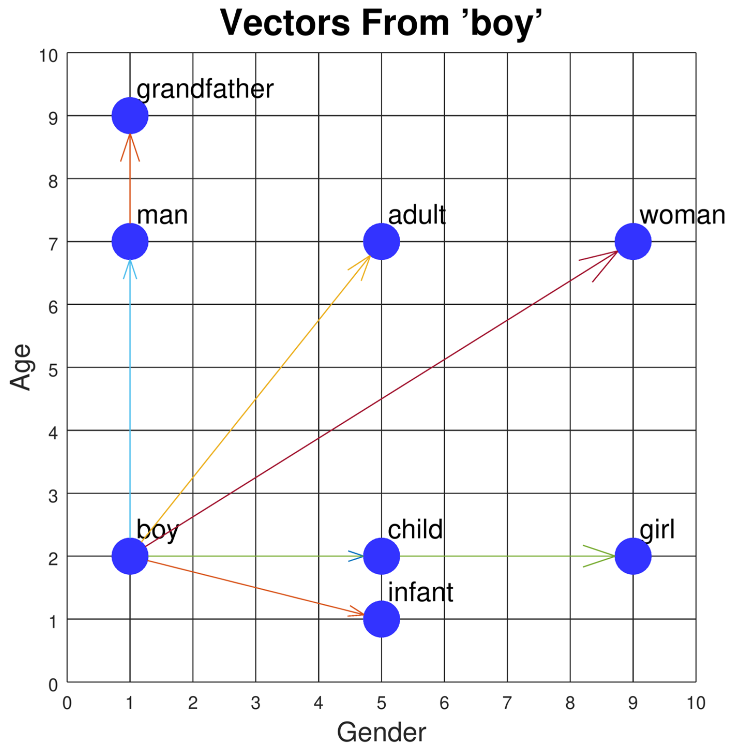
Scott

词嵌入

- embedding

word embedding: ”embedding”: [[-0.006929283495992422, -0.005336422007530928, … -4.547132266452536e-05, -0.024047505110502243] ,…,]

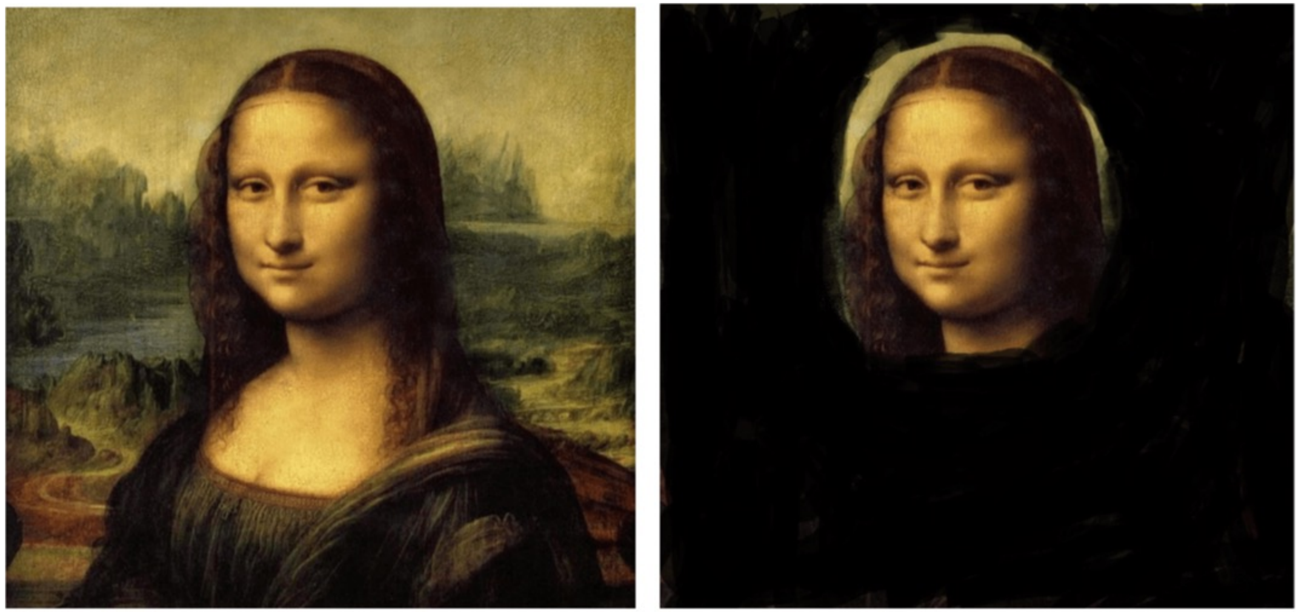
Word2vec
GloVe
ELMO
BERT
Sentence-BERT



向量空间

https://www.cs.cmu.edu/~dst/WordEmbeddingDemo/tutorial.html

Attention



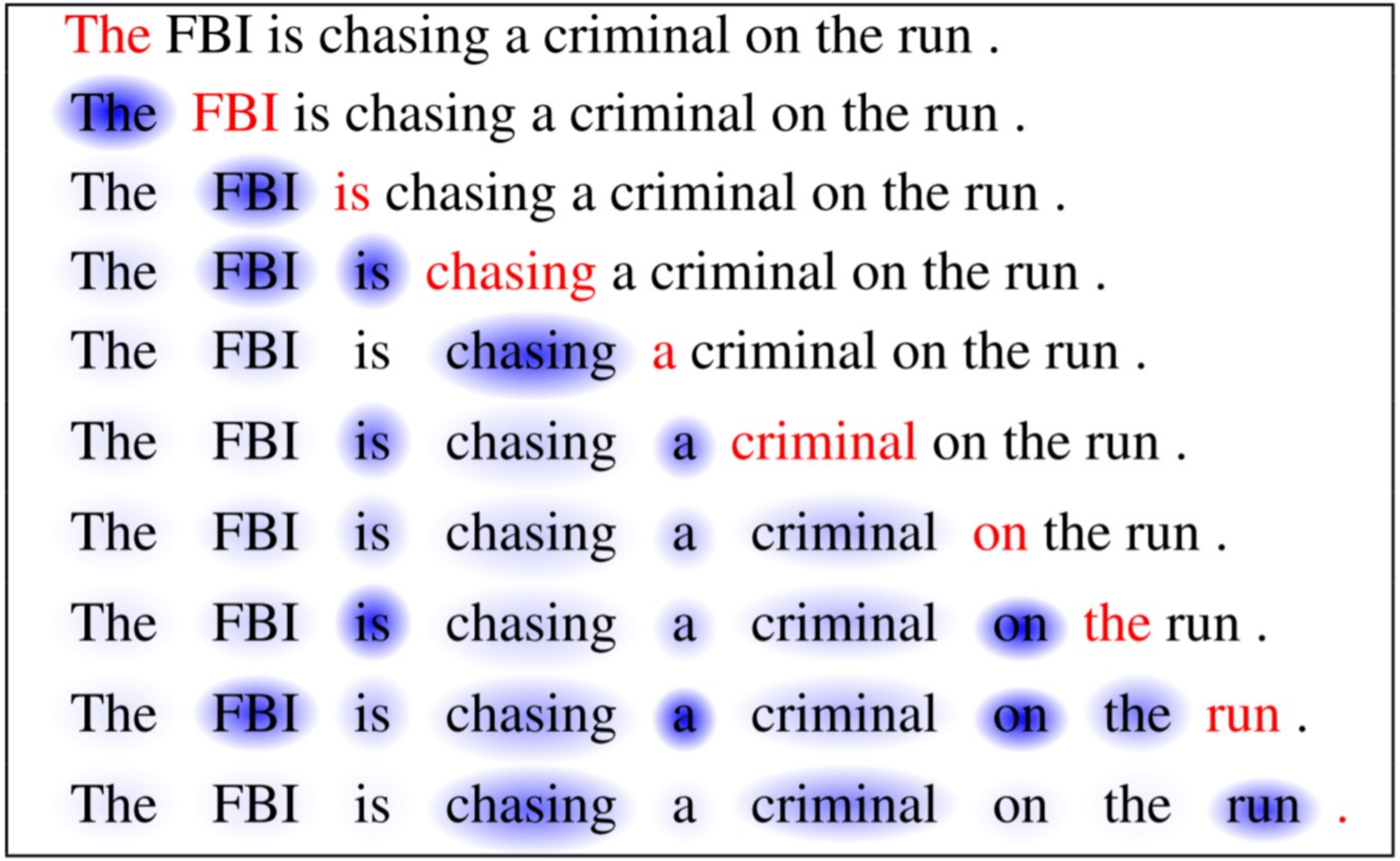
(a)

Mona Lisa is a portrait painted by Leonardo da Vinci.

Mona Lisa is a portrait painted by Leonardo da Vinci.

(b)

Similarity



inner product: $s_i = \boldsymbol{q}^\top \boldsymbol{k}_i,$

scaled inner product: $s_i = \dfrac{\boldsymbol{q}^\top \boldsymbol{k}_i}{\sqrt{p}},$

general inner product: $s_i = \boldsymbol{q}^\top \boldsymbol{W} \boldsymbol{k}_i,$

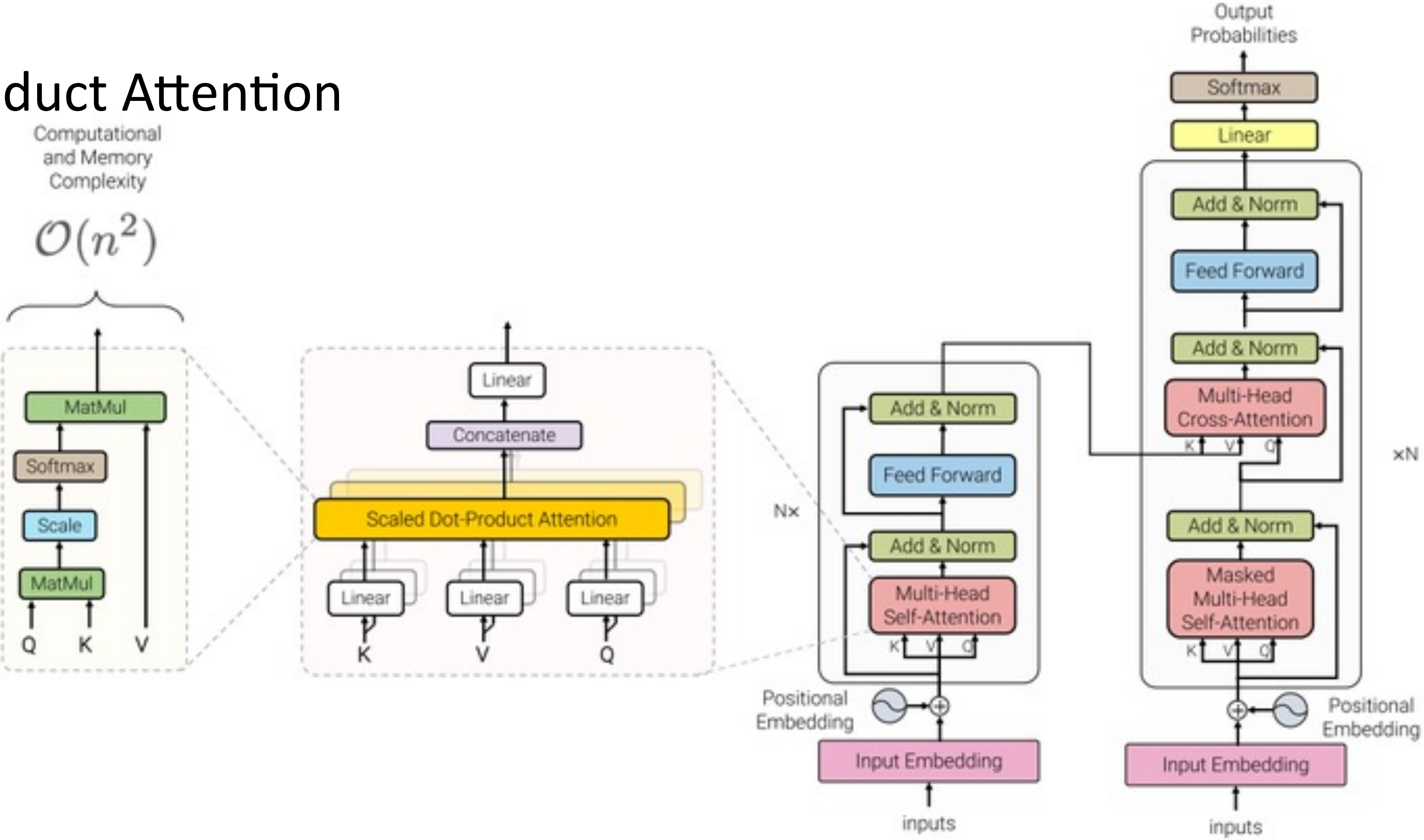
additive similarity: $s_i = \boldsymbol{w}_q^\top \boldsymbol{q} + \boldsymbol{w}_k^\top \boldsymbol{k}_i,$

Additive Attention
Dot-product Attention
Scaled Dot-product Attention

Cross-attention
Self-attention
Masked attention

Transformer结构

- Scaled-dot-product Attention
- Multi-head
- Position Encoding
- Layer Norm



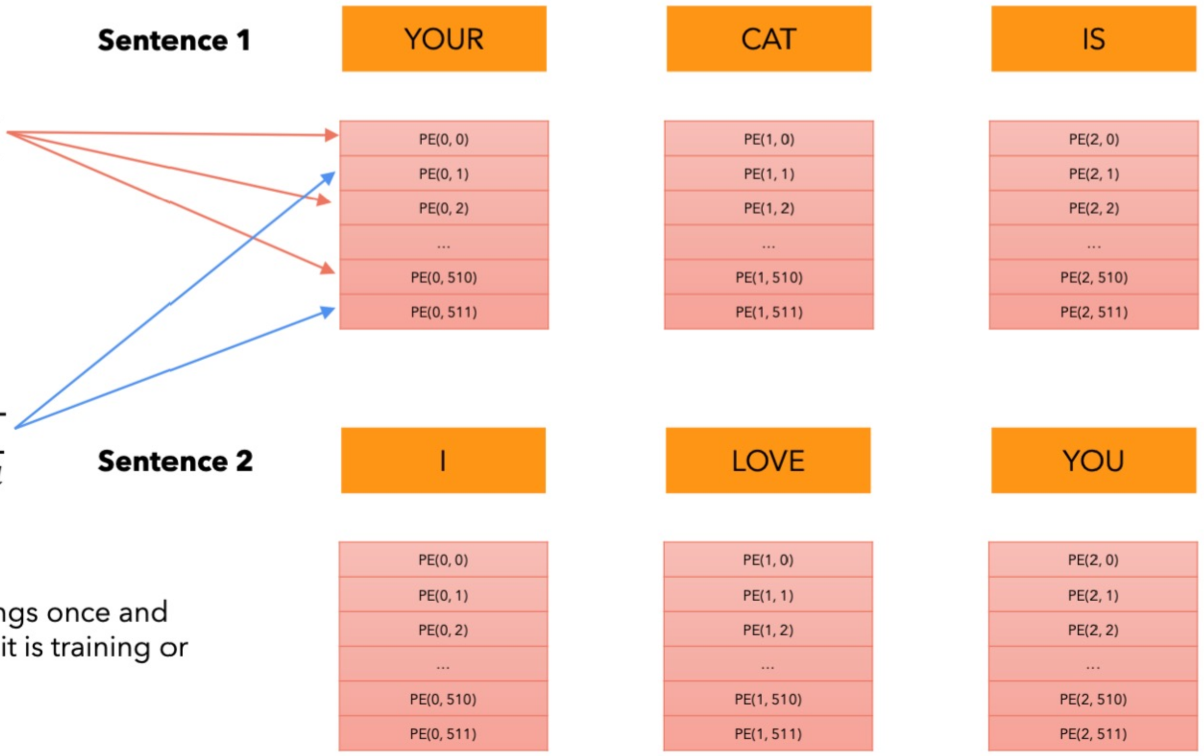
Input Embedding + Position Embedding

Original sentence	YOUR	CAT	IS	A	LOVELY	CAT
Embedding (vector of size 512)	952.207	171.411	621.659	776.562	6422.693	171.411
	5450.840	3276.350	1304.051	5567.288	6315.080	3276.350
	1853.448	9192.819	0.565	58.942	9358.778	9192.819
	…	…	…	…	…	…
	1.658	3633.421	7679.805	2716.194	2141.081	3633.421
	2671.529	8390.473	4506.025	5119.949	735.147	8390.473
	+	+	+	+	+	+
Position Embedding (vector of size 512). Only computed once and reused for every sentence during training and inference.	…	1664.068	…	…	…	1281.458
	…	8080.133	…	…	…	7902.890
	…	2620.399	…	…	…	912.970
	…	…	…	…	…	3821.102
	…	9386.405	…	…	…	1659.217
	…	3120.159	…	…	…	7018.620
	=	=	=	=	=	=
Encoder Input (vector of size 512)	…	1835.479	…	…	…	1452.869
	…	11356.483	…	…	…	11179.24
	…	11813.218	…	…	…	10105.789
	…	…	…	…	…	…
	…	13019.826	…	…	…	5292.638
	…	11510.632	…	…	…	15409.093

Absolute Position Embedding

$$PE(pos, 2i) = \sin \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$

$$PE(pos, 2i+1) = \cos \frac{pos}{10000^{\frac{2i}{d_{model}}}}$$



We only need to compute the positional encodings once and then reuse them for every sentence, no matter if it is training or inference.

Umar Jamil - https://github.com/hkproj/transformer-from-scratch-notes

Attention Kernel

Self-Attention allows the model to relate words to each other.

In this simple case we consider the sequence length **seq** = 6 and $\mathbf{d_{model}} = \mathbf{d_k} = 512$.

The matrices **Q**, **K** and **V** are just the input sentence.

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

$$\text{softmax}\left(\frac{\underset{(6,\,512)}{\mathbf{Q}} \times \underset{(512,\,6)}{\mathbf{K^T}}}{\sqrt{512}}\right) =$$

	YOUR	CAT	IS	A	LOVELY	CAT	Σ
YOUR	0.268	0.119	0.134	0.148	0.179	0.152	1
CAT	0.124	0.278	0.201	0.128	0.154	0.115	1
IS	0.147	0.132	0.262	0.097	0.218	0.145	1
A	0.210	0.128	0.206	0.212	0.119	0.125	1
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174	1
CAT	0.195	0.114	0.203	0.103	0.157	0.229	1

(6, 6)

* all values are random.

* for simplicity I considered only one head, which makes $d_{model} = d_k$.

Scaled-dot Attention

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

	YOUR	CAT	IS	A	LOVELY	CAT
YOUR	0.268	0.119	0.134	0.148	0.179	0.152
CAT	0.124	0.278	0.201	0.128	0.154	0.115
IS	0.147	0.132	0.262	0.097	0.218	0.145
A	0.210	0.128	0.206	0.212	0.119	0.125
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174
CAT	0.195	0.114	0.203	0.103	0.157	0.229

(6, 6)

X

V

(6, 512)

=

Attention

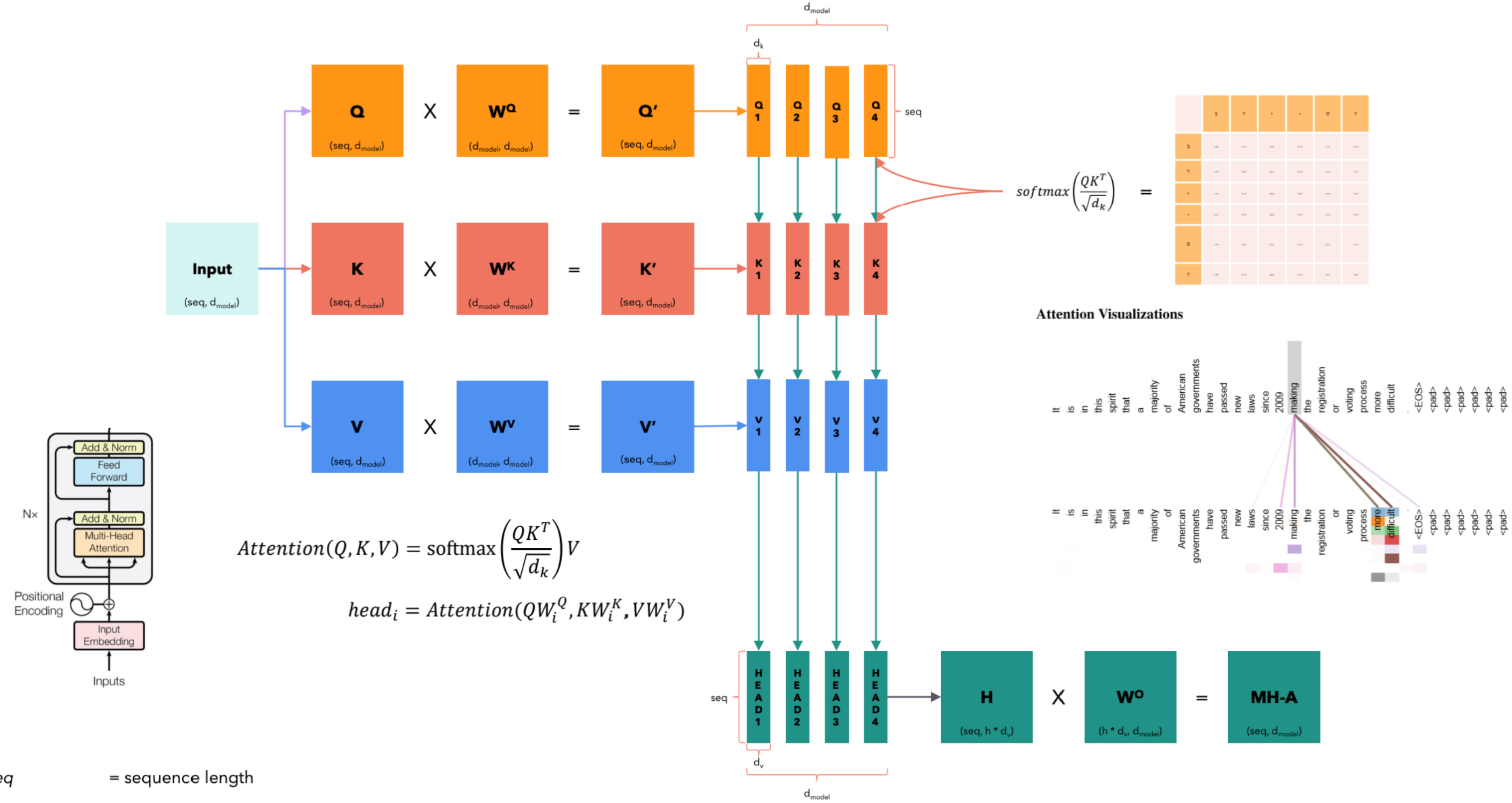
(6, 512)

Each row in this matrix captures not only the meaning (given by the embedding) or the position in the sentence (represented by the positional encodings) but also each word's interaction with other words.

Why attention

	YOUR	CAT	IS	A	LOVELY	CAT
YOUR	0.268	0.119	0.134	0.148	0.179	0.152
CAT	0.124	0.278	0.201	0.128	0.154	0.115
IS	0.147	0.132	0.262	0.097	0.218	0.145
A	0.210	0.128	0.206	0.212	0.119	0.125
LOVELY	0.146	0.158	0.152	0.143	0.227	0.174
CAT	0.195	0.114	0.203	0.103	0.157	0.229

Multi-head Attention



$$softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) =$$

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

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

$$MultiHead(Q, K, V) = Concat(head_1 \; \ldots head_h)W^O$$

seq = sequence length

d_{model} = size of the embedding vector

h = number of heads

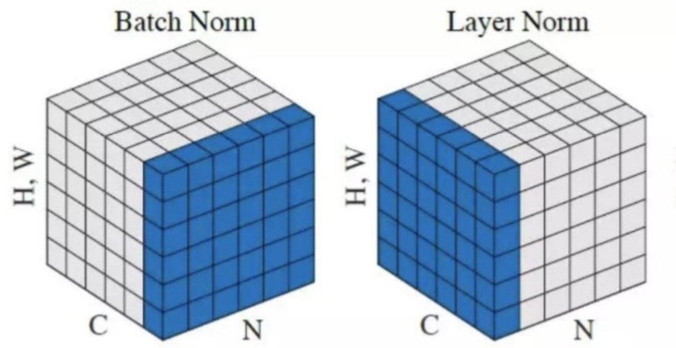
$d_k = d_v$ = d_{model} / h

Umar Jamil – https://github.com/hkproj/transformer-from-scratch-notes

Layer Norm

Batch of 3 items

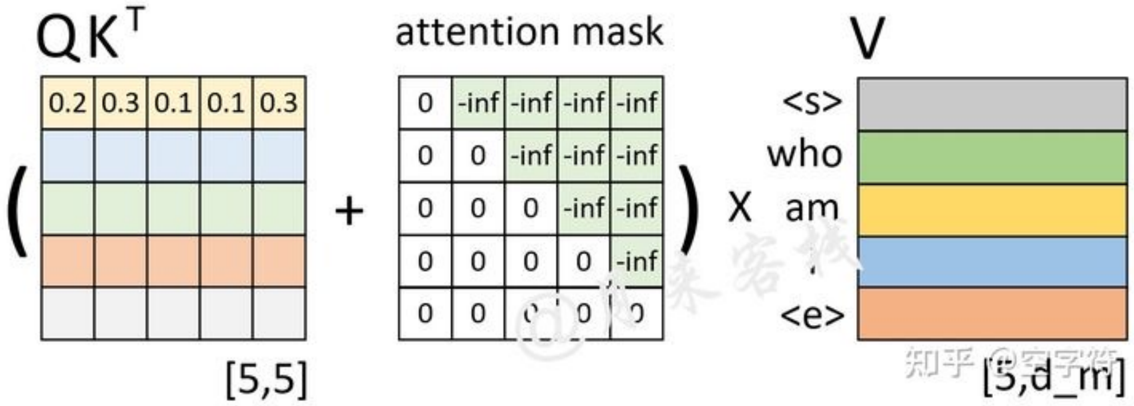
ITEM 1	ITEM 2	ITEM 3
50.147	1242.223	9.370
3314.825	688.123	4606.674
...	...	...
...	...	...
8463.361	434.944	944.705
8.021	149.442	21189.444
μ_1	μ_2	μ_3
σ_1^2	σ_2^2	σ_3^2



$$\hat{x}_j = \frac{x_j - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$

We also introduce two parameters, usually called **gamma** (multiplicative) and **beta** (additive) that introduce some fluctuations in the data, because maybe having all values between 0 and 1 may be too restrictive for the network. The network will learn to tune these two parameters to introduce fluctuations when necessary.

Masked Multi-head Attention

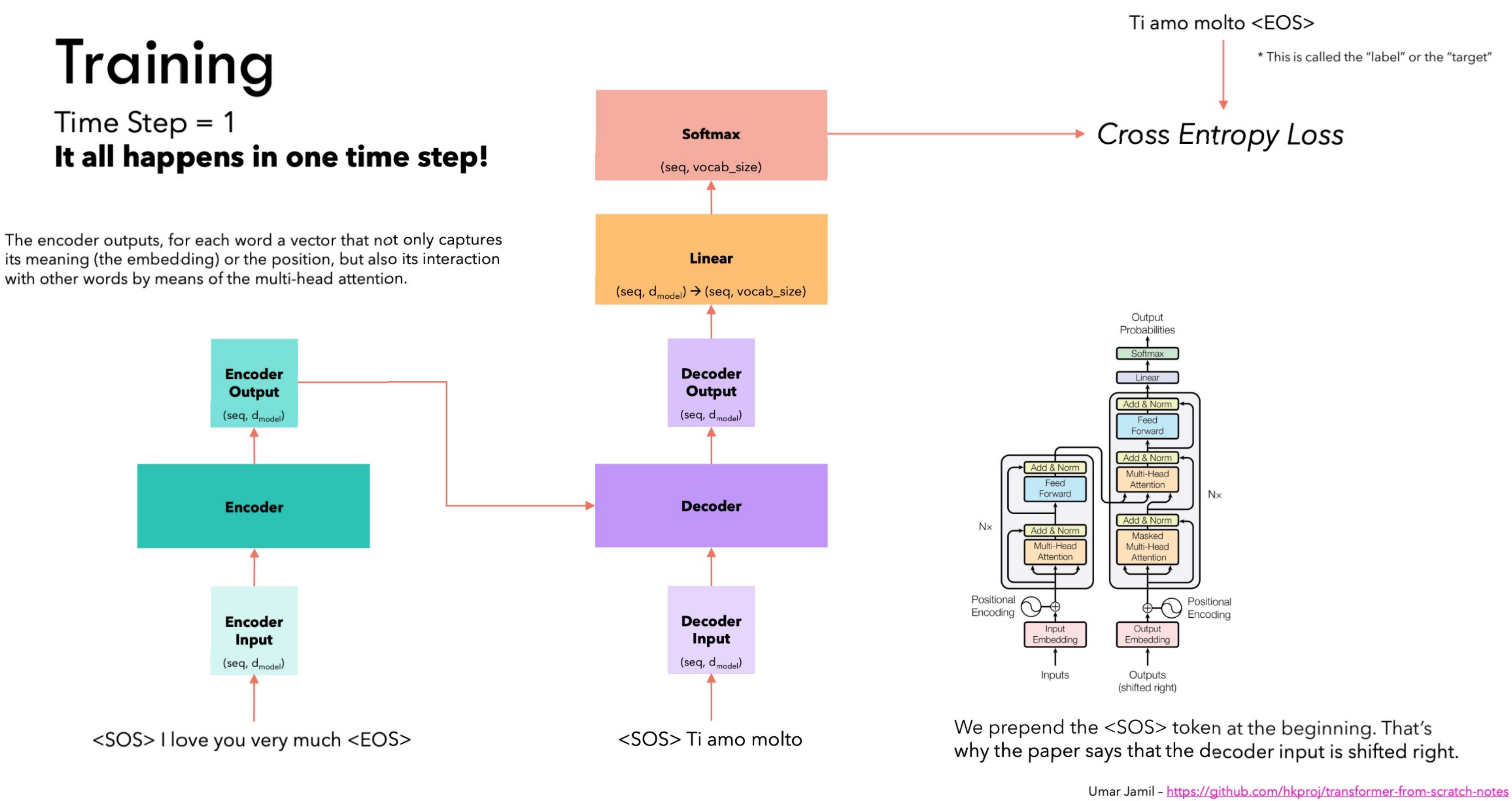


	A	B	C	D
A	0.11	-inf	-inf	-inf
B	0.19	0.50	-inf	-inf
C	0.53	0.98	0.95	-inf
D	0.81	0.86	0.38	0.90

Masked

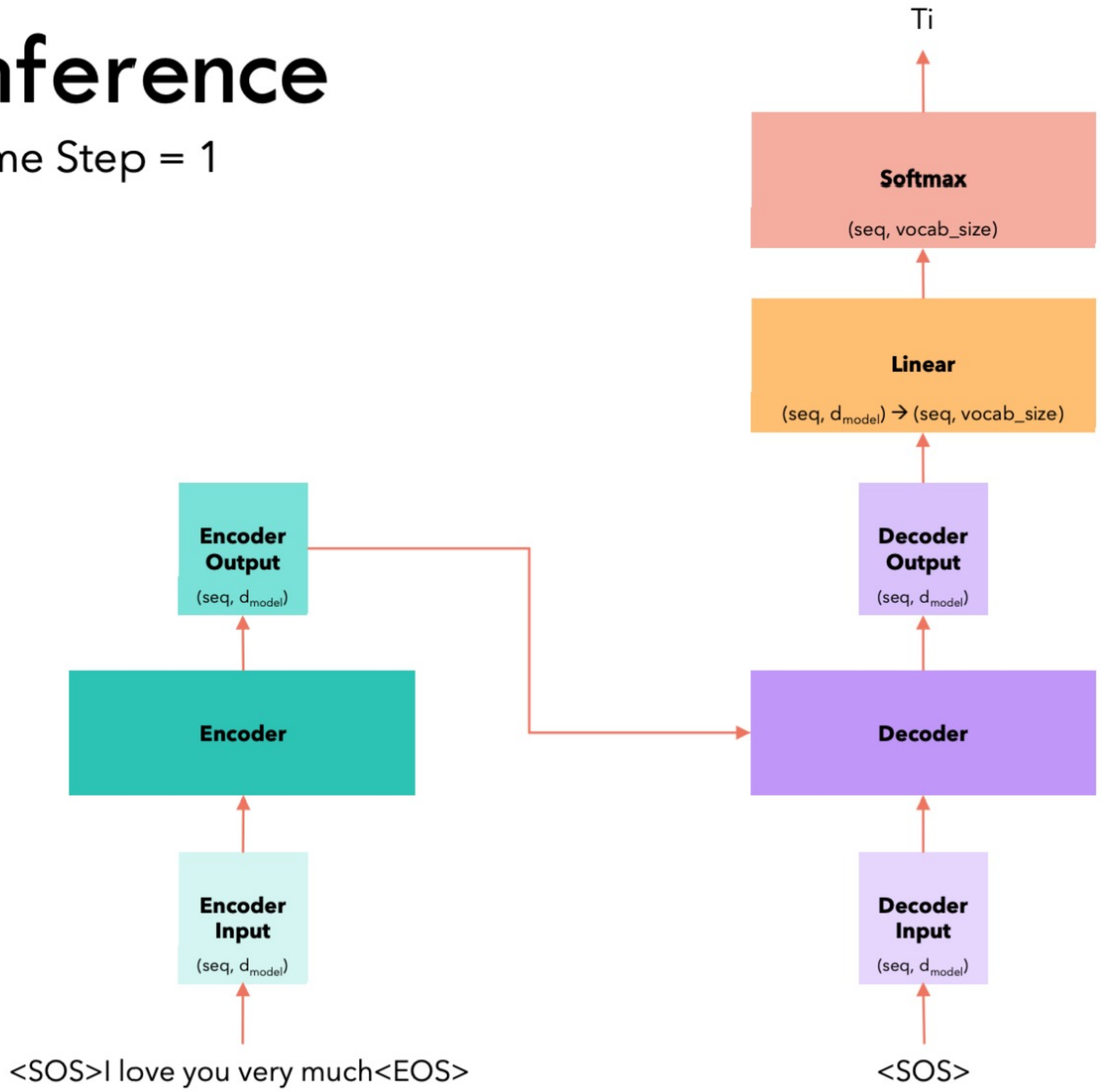
	A	B	C	D
A	1	0	0	0
B	0.48	0.52	0	0
C	0.31	0.35	0.34	0
D	0.25	0.26	0.23	0.26

Softmax


Training
Time Step = 1
It all happens in one time step!
The encoder outputs, for each word a vector that not only captures its meaning (the embedding) or the position, but also its interaction with other words by means of the multi-head attention.
Ti amo molto <EOS>
* This is called the "label" or the "target"
Cross Entropy Loss
Softmax
(seq, vocab_size)
Linear
(seq, d_{model}) → (seq, vocab_size)
Encoder Output
(seq, d_{model})
Decoder Output
(seq, d_{model})
Encoder
Decoder
Encoder Input
(seq, d_{model})
Decoder Input
(seq, d_{model})
<SOS> I love you very much <EOS>
<SOS> Ti amo molto
Output Probabilities
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
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)
We prepend the <SOS> token at the beginning. That's why the paper says that the decoder input is shifted right.
Umar Jamil – https://github.com/hkproj/transformer-from-scratch-notes

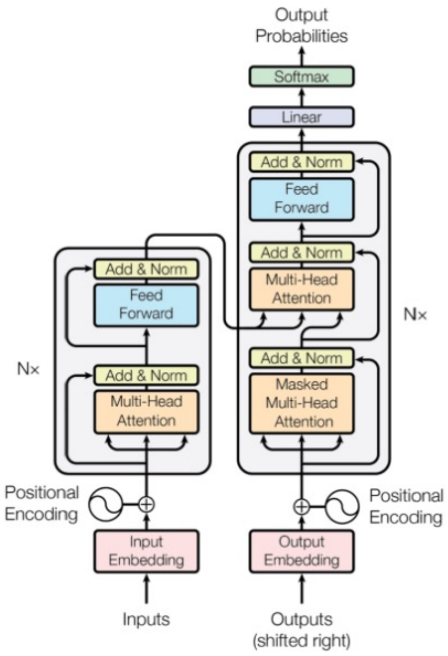
Inference

Time Step = 1



We select a token from the vocabulary corresponding to the position of the token with the maximum value.

The output of the last layer is commonly known as **logits**



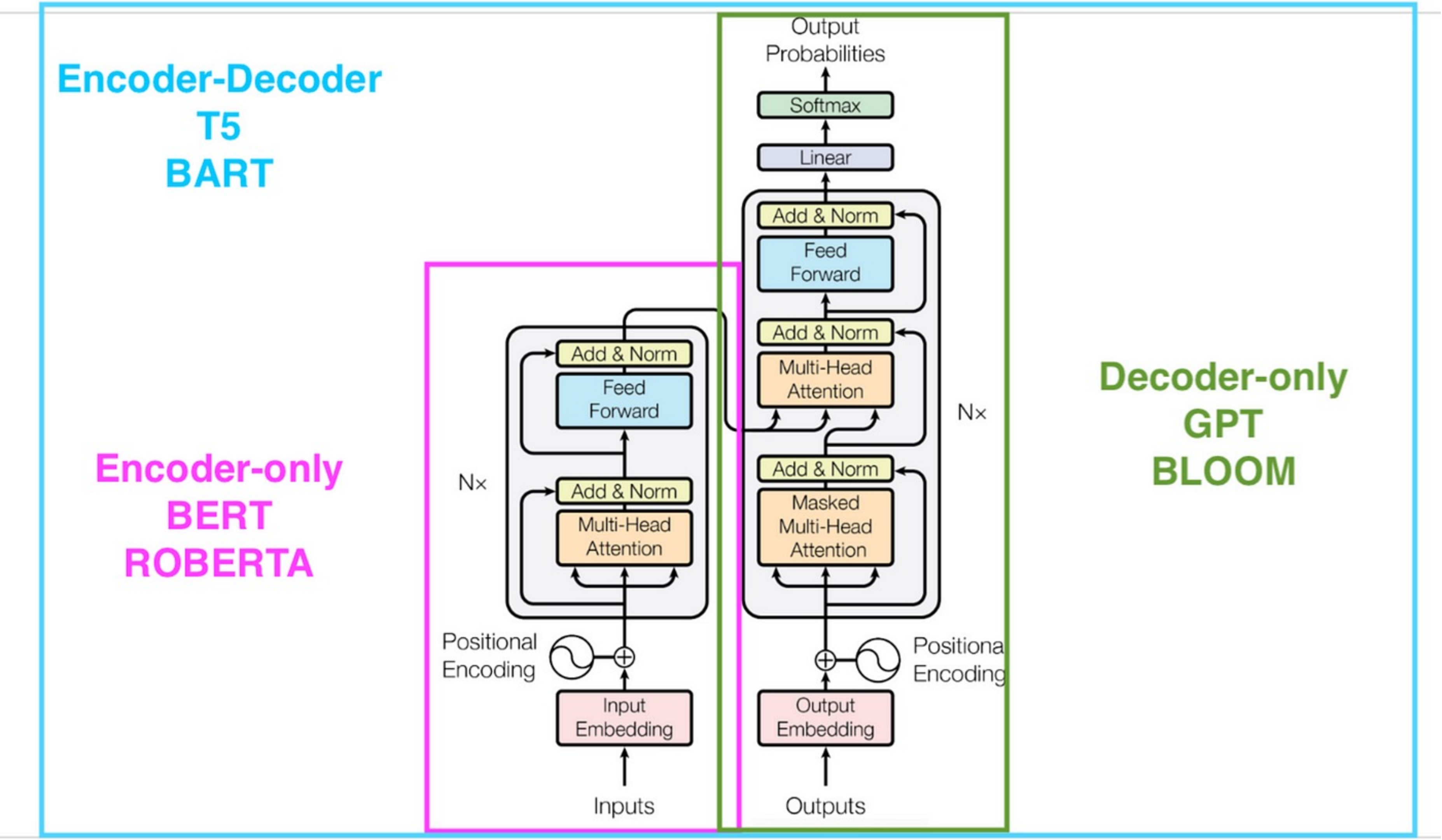
* Both sequences will have same length thanks to padding

Umar Jamil - https://github.com/hkproj/transformer-from-scratch-notes

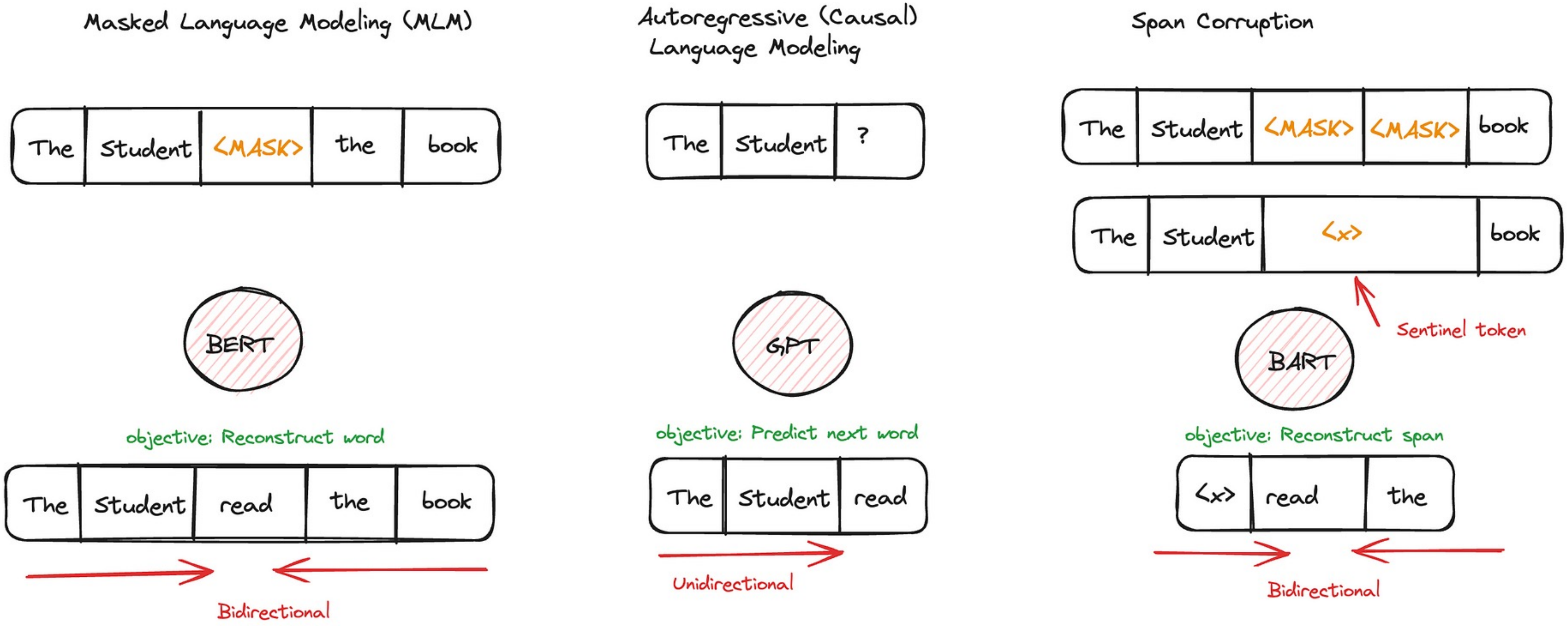
BERT / GPT / BART

	BERT	GPT	BART
Model Type	Encoder Only	Decoder Only	Encoder-Decoder
Direction	Bidirectional	Unidirectional (left-to-right)	Bidirectional
Pre-training Objective	Masked language modeling (MLM)	Autoregressive (casual) language modeling	Span Corruption (Masking entire spans of words)
Fine-tuning	Task-specific layer added on top of the pre-trained BERT model	Providing task-specific prompts using few-shot or one-shot adaptation and adapting the model's parameters	Versatile and can be used for various NLP tasks
Use Case	Sentiment Analysis Named entity Recognition Word Classification	Text generation Text completion creative writing	Translation Text Summarisation Question & Answer
Original Organisations	Google AI	OpenAI	Facebook AI

BERT / GPT / BART



BERT / GPT / BART



原文

Attention Is All You Need

Ashish Vaswani[*] Google Brain avaswani@google.com

Noam Shazeer[*] Google Brain noam@google.com

Niki Parmar[*] Google Research nikip@google.com

Jakob Uszkoreit[*] Google Research usz@google.com

Llion Jones[*] Google Research llion@google.com

Aidan N. Gomez[*] [†] University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser[*] Google Brain lukaszkaiser@google.com

Illia Polosukhin[*] [‡] illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

[*]Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

[†]Work performed while at Google Brain.

[‡]Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

智能体 Agent

- **Chain-of-Verification Reduces Hallucination in Large Language Models.** *Shehzaad Dhuliawala (Meta AI & ETH Zu¨rich) et al. arXiv.* [paper]
- **SelfCheck: Using LLMs to Zero-Shot Check Their Own Step-by-Step Reasoning.** *Ning Miao (University of Oxford) et al. arXiv.* [paper] [code]
- **ChatCoT: Tool-Augmented Chain-of-Thought Reasoning on Chat-based Large Language Models.** *Zhipeng Chen (Renmin University of China) et al. arXiv.* [paper] [code]
- Improving Factuality and Reasoning in Language Models through Multiagent Debate
- Igniting Language Intelligence: The Hitchhiker's Guide From Chain-of-Thought Reasoning to Language Agents