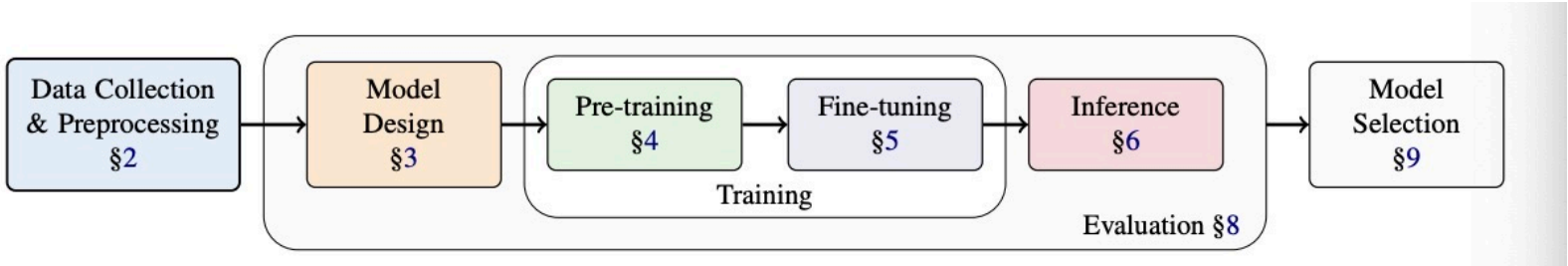
Transformer架构

Scott

回顾

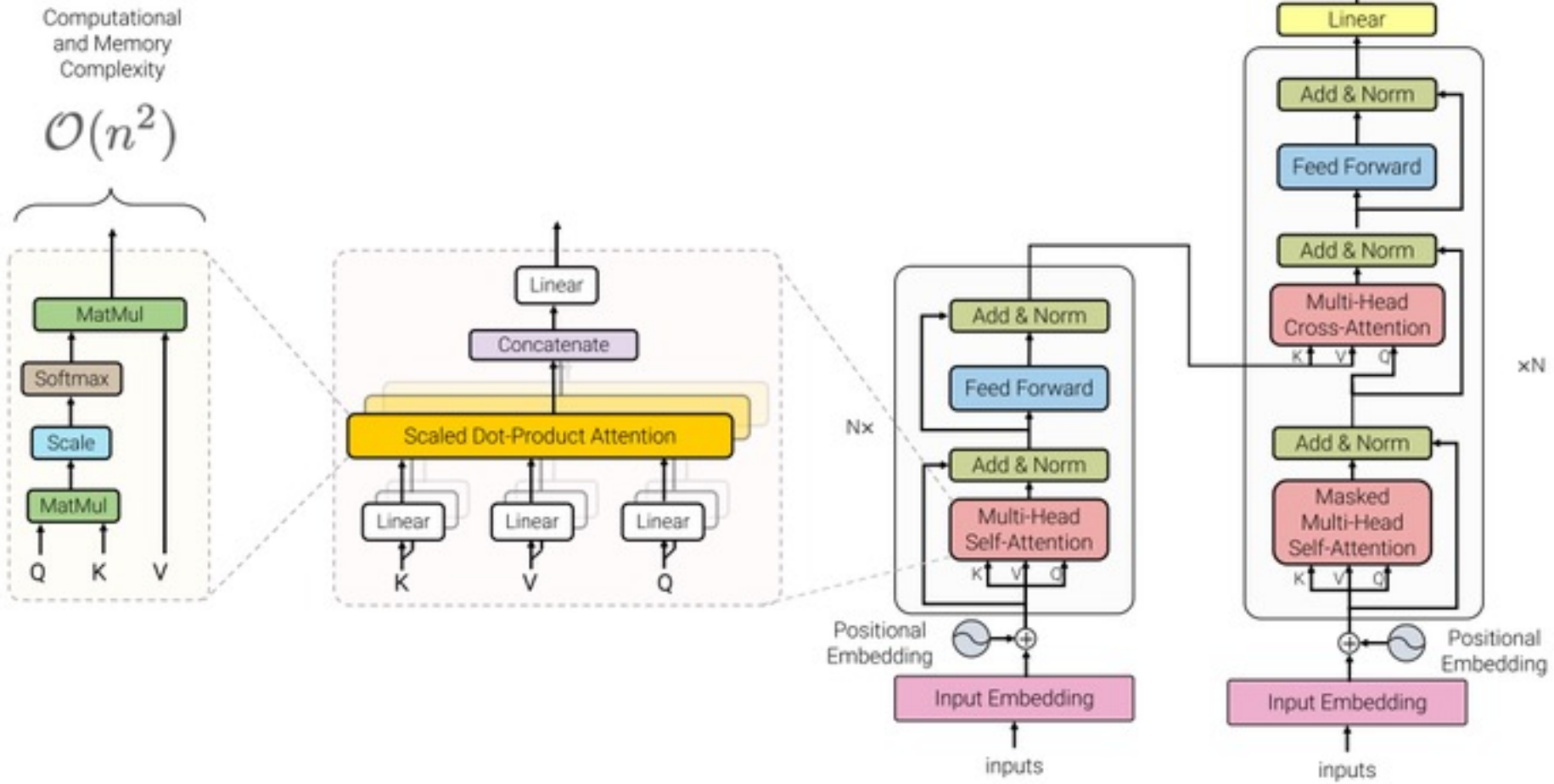
- 自然语言处理发展进程（后transformer时代）范式转移与统一范式



自然语言推理
The boy is sleeping
The boy is resting

Transformer结构

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



编码与解码

- Bytes / str

英文：deepshare
中文：深度之眼
字节流：

b’\xe6\xb7\xb1\xe5\xba\xa6\xe4\xb9\x8b\xe7\x9c\xbc’ UTF-8
b‘\\u6df1\\u5ea6\\u4e4b\\u773c’ UNICODE
b'\xc9\xee\xb6\xc8\xd6\xae\xd1\xdb‘ GBK

```
1  def convert_to_unicode(text):
2      """Converts `text` to Unicode (if it's not already), assuming utf-8 input."""
3      if isinstance(text, str):
4        return text
5      elif isinstance(text, bytes):
6        return text.decode("utf-8", "ignore")
7      else:
8        raise ValueError("Unsupported string type: %s" % (type(text)))
```

分词（令牌化）

- tokenize

Tokenize(令牌): [CLS] The boy is sleeping . [SEP]
input ids: [101, 464, 2933, 318, 11029, 13, 102]

BasicTokenizer
转成 unicode -> 去除各种奇怪字符 -> 处理中文 -> 空格分词 -> 去除多余字符和标点分词 -> 再次空格分词

WordpieceTokenizer
转成Unicode->空格分词->异常词处理->加载词典->匹配算法

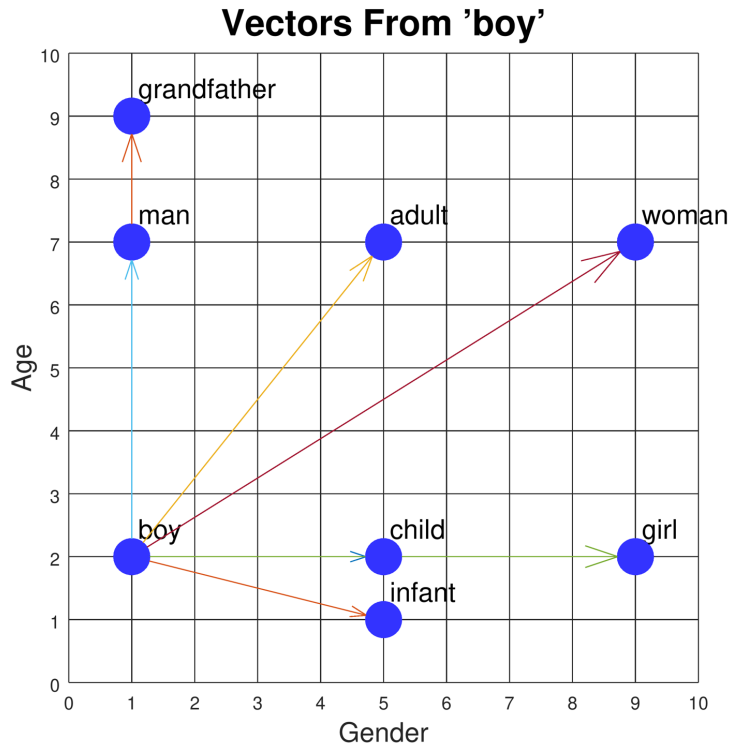
```
1  def convert_by_vocab(vocab, items):
2    """Converts a sequence of [tokens|ids] using the vocab."""
3    output = []
4    for item in items:
5      output.append(vocab[item])
6    return output
```

词嵌入

- embedding

word embedding: ”embedding”: [[-0.00692928349599242, -0.0053364220075309 28, ... -4.547132266452536e-05, -0.02404750511050224 3] ,...,]

Word2vec
GloVe
ELMO
BERT
Sentence-BERT



向量空间

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

词典

1	[PAD]	101	[UNK]	1000	!										
2	[unused0]	102	[CLS]	1001	"	1015	0	1038	a	1736	ロ	1997	the		
3	[unused1]	103	[SEP]	1002	#	1016	1	1039	b	1737	ワ	1998	of		
4	[unused2]	104	[MASK]	1003	$	1017	2	1040	c	1738	ン	1999	and		
5	[unused3]	105	[unused99]	1004	%	1018	3	1041	d	1739	·	2000	in		
6	[unused4]	106	[unused100]	1005	&	1019	4	1042	e	1740	ー	2001	to		
7	[unused5]	107	[unused101]	1006	'	1020	5	1043	f	1741	一	2002	was		
8	[unused6]	108	[unused102]	1007	(	1021	6	1044	g	1742	三	2003	he		
9	[unused7]	109	[unused103]	1008	)	1022	7	1045	h	1743	上	2004	is		
10	[unused8]	110	[unused104]	1009	*	1023	8	1046	i	1744	下	2005	as		
11	[unused9]	111	[unused105]	1010	+	1024	9	1047	j	1745	不	2006	for		
12	[unused10]			1011	,										

Input Embedding + Position Embedding

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

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.

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

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

	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)

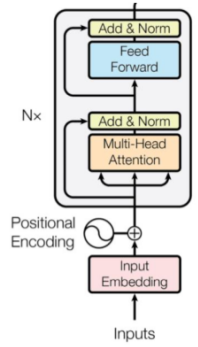
Each row in this matrix captures not only the meaning (given by the embedding) or the position in the sentence (represented by the positional encodings) but also each word's interaction with other words.

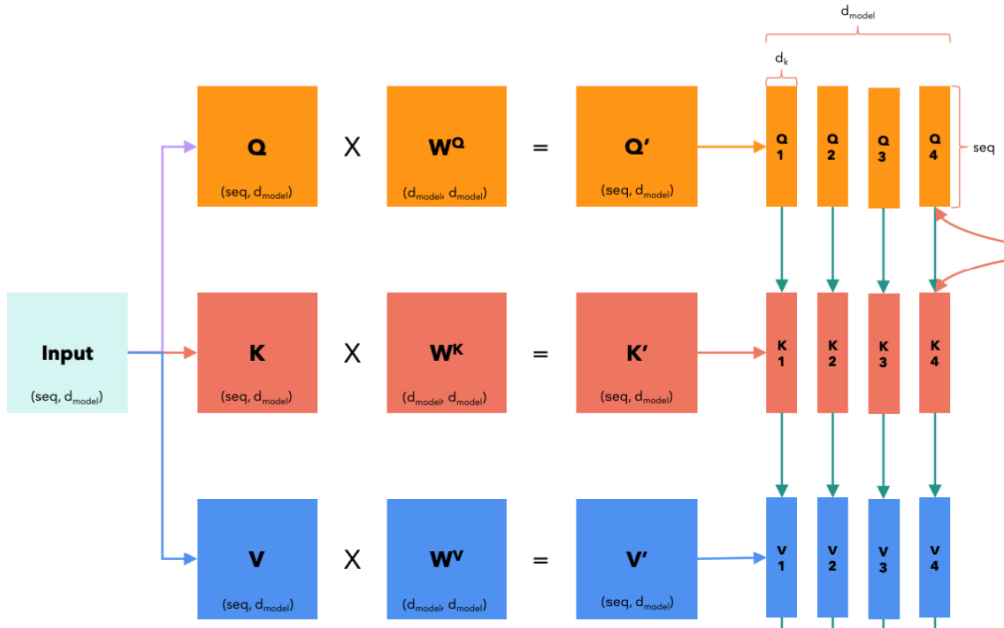
Umar Jamil - https://github.com/hkproj/transformer-from-scratch-notes

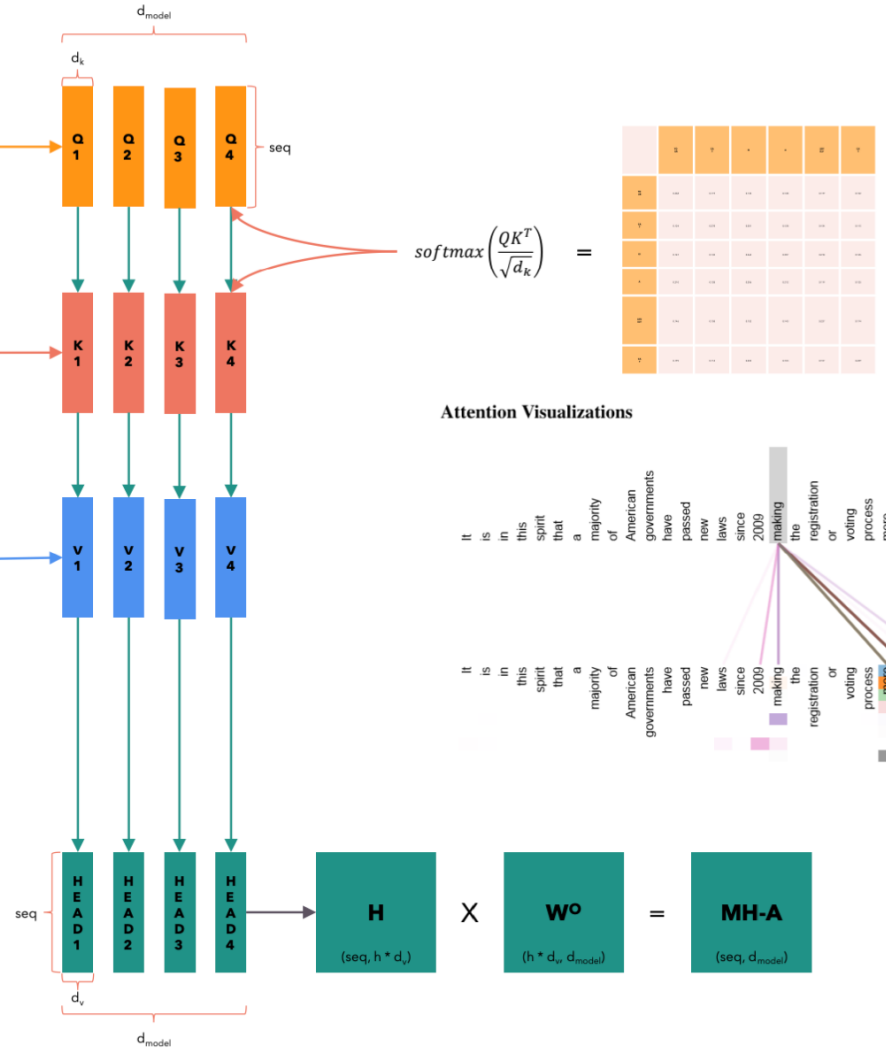
Why attention

	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 Visualizations

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

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

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

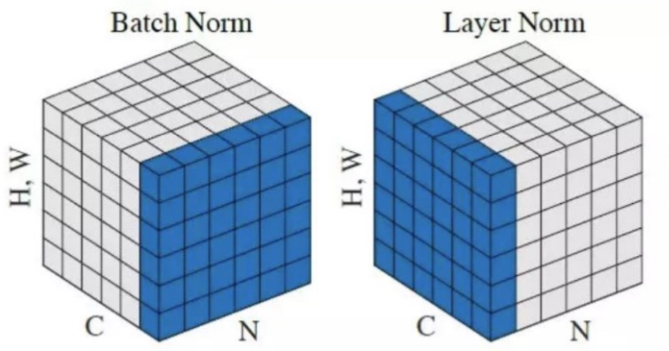
seq	= sequence length
d_{model}	= size of the embedding vector
h	= number of heads
$d_k = d_v$	= d_{model} / h

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

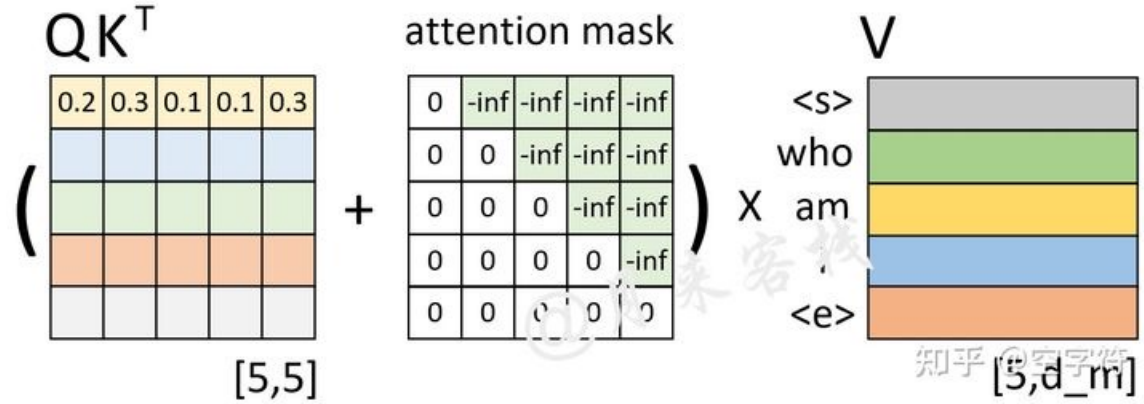


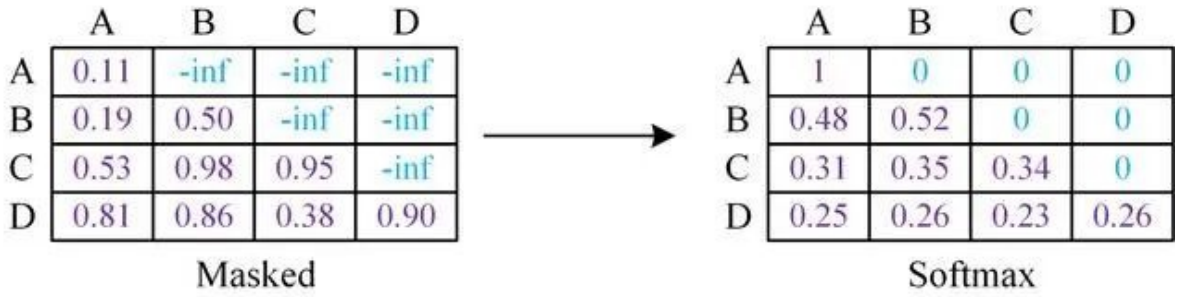
$$\hat{x}_j = \frac{x_j - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$

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.

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

Masked Multi-head Attention





Training

Time Step = 1

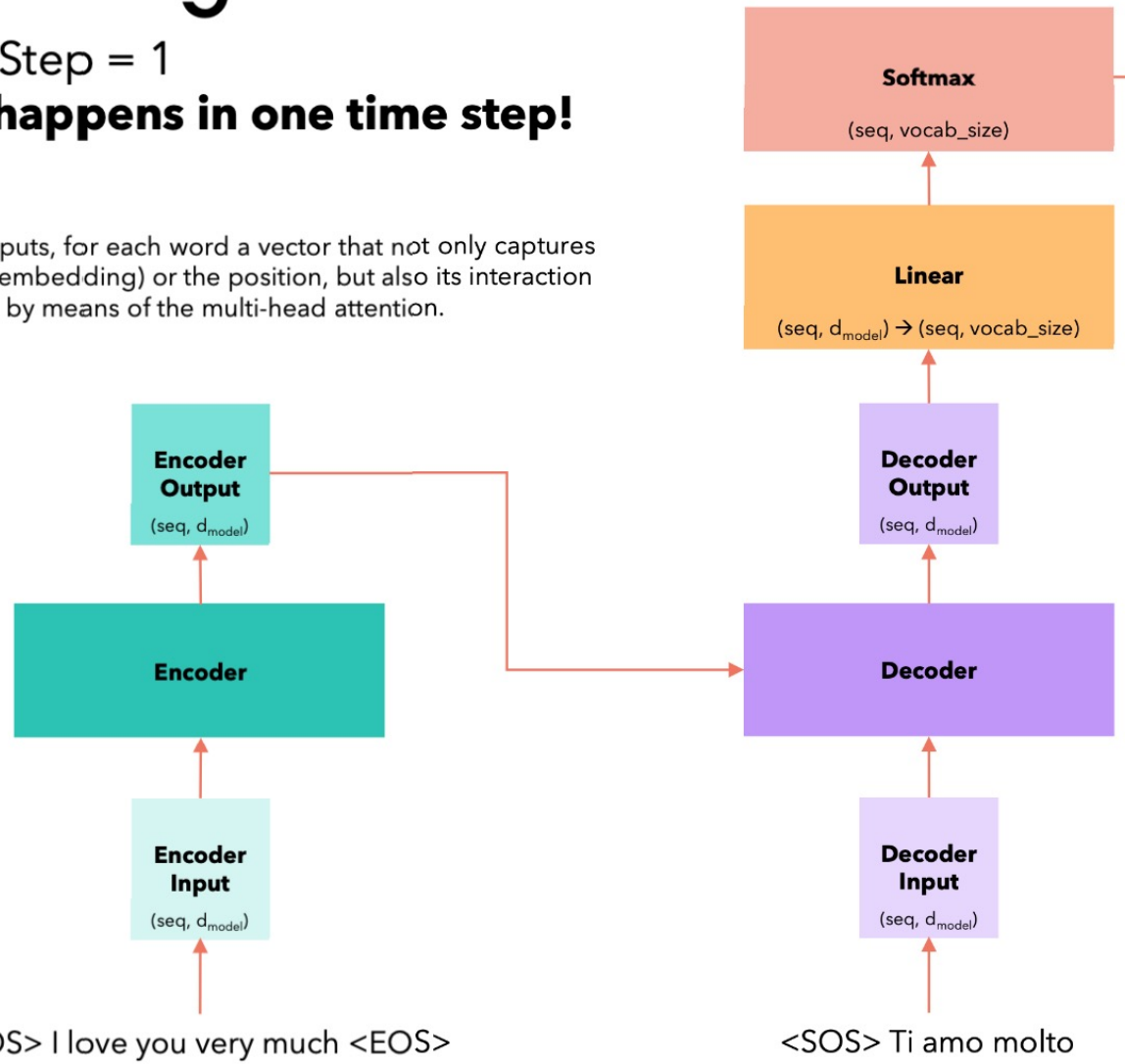
It all happens in one time step!

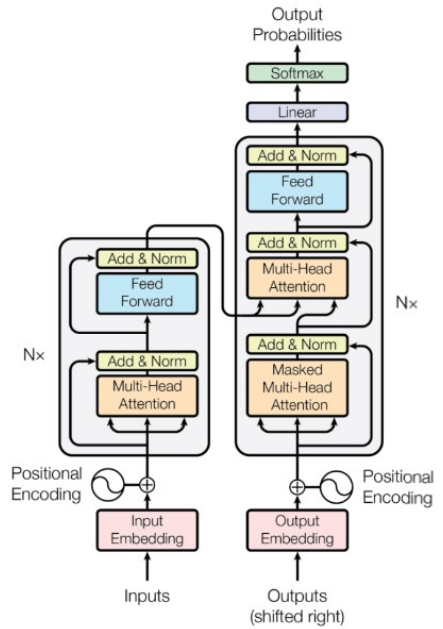
Ti amo molto <EOS>

* This is called the "label" or the "target"

Cross Entropy Loss

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.



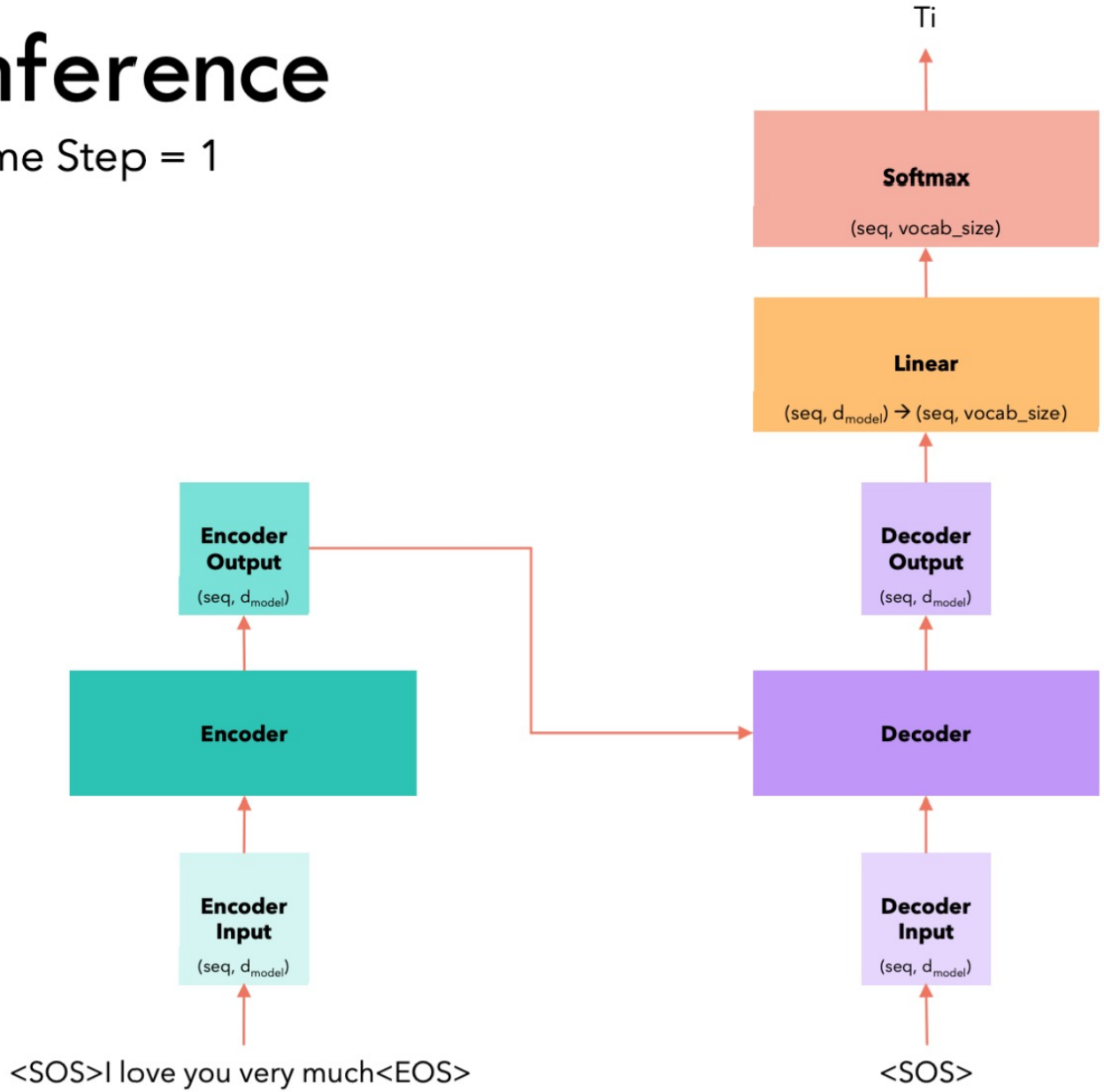


We prepend the <SOS> token at the beginning. That's why the paper says that the decoder input is shifted right.

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

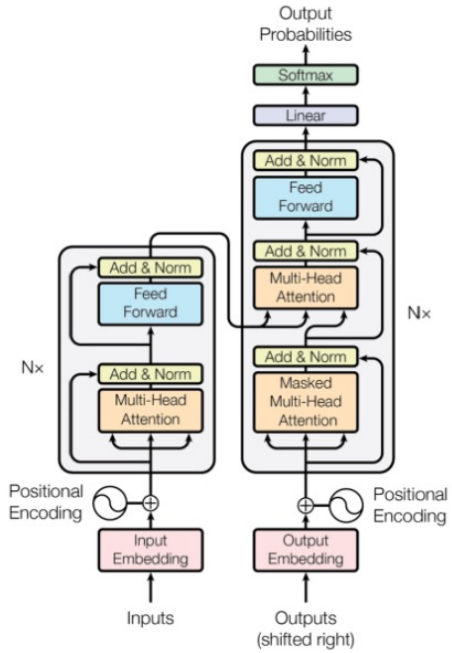
Inference

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

OpenAI账号

- ChatGPT
- API
- `openai` python lib
- `openeval` python lib

```
`pip install openai==0.28`

export OPENAI_API_KEY=“”
```

模型下载加速

- Huggingface模型下载

1. 自动下载 ./cache
2. Huggingface hub

镜像 + hf_transfer

Github镜像

githubfast.com

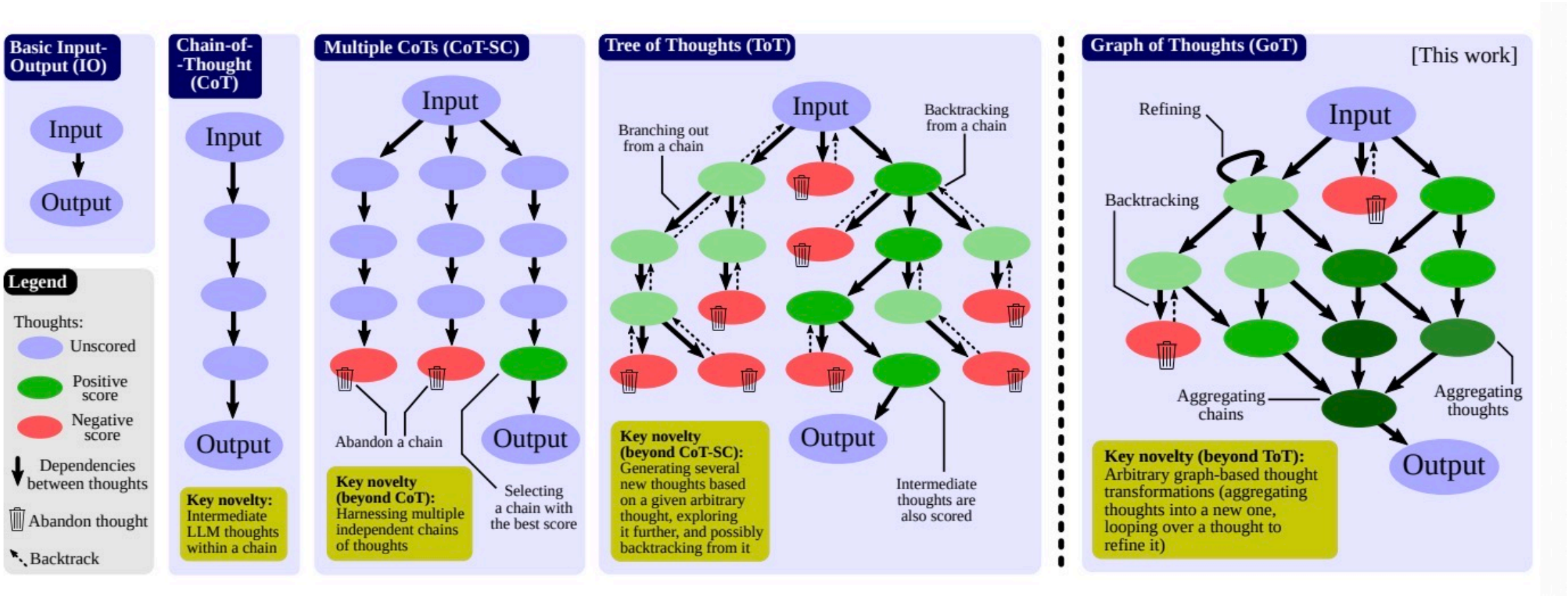
pip install -U huggingface_hub

export HF_ENDPOINT="https://hf-mirror.com"

export HF_HUB_ENABLE_HF_TRANSFER=1

huggingface-cli download --token hf_*** --resume-download meta-llama/Llama-2-7b-hf --local-dir Llama-2-7b-hf

Few-shot/Zero-shot CoT. Self-consistency. ToT



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