



计算机系 宋曦轩

<https://github.com/sast-summer-training-2023/sast2023-nlp>

自然语言处理

- 如何表示一个词的含义？
- 如何获取词在句子中的向量表示？
- 如何在模型中存储知识？
- 如何基于以上原理构建语言模型？

🤗 如何表示一个词的含义？

- 用一个id表示一个词？

北京 🐳 = 344
南京 🐳 = 345
中学 = 364
大学 = 365
北大 = 344 + 365 = 709?

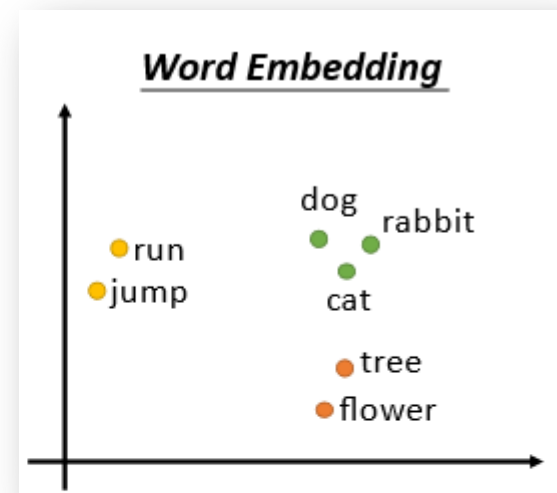
- 用一个向量表示一个词

北京 🐳 = [0.1, 0, 0.9, 0, 0, 0]
南京 🐳 = [0.3, 0, 0.9, 1, 0, 0]
中学 = [0, 0, 0, 0, 0.5, 1]
大学 = [0, 0, 0, 0, 0.9, 1]
北大 = [0.1, 0, 0.9, 0, 0.9, 1]

🤗 Word Embedding

- 将一个词映射为一个 *embedding* dim 维的向量
- 每一维具有一定的含义（具体含义可能很抽象）

维度	金属	生物	👋	🤗
电子羊	0.9	0.1	0.3	0.2
仿生人	0.5	0.4	0.4	0.8
拔罐王	0.2	0.8	0.9	1.0



🤗 如何获取 Word Embedding ?

- 手工构造?
 - “北京在北纬40度，所以必须有一维向量的值是40”？
- 通过在任务中拟合数据获得
 - 例如对于 Next token prediction（根据前缀预测下一个词）的任务：
 1. 随机初始化所有词 v 的 Word Embedding E_v
 2. 将前缀的 Embedding 求和取平均 $E_{predict} = \frac{1}{k} \sum_{j=0}^{k-1} E_{v_j}$
 3. 下一个词是 Embedding 与 $E_{predict}$ 最接近的词
$$v_{predict} = \max_{v_i} \cos \langle E_{predict}, E_{v_i} \rangle$$
 4. 计算 $loss$, 反向传播

🤗 如何获取词在句子中的向量表示？

- **An Apple a Day Keeps the Doctor Away**
- 一天一部 iPhone 让我与博士学位失之交臂？
- 🍏 or 🍏?
- 有没有被咬一口？
- 一个简单的想法：
- 将句中所有词的向量加权求和，表示词在句中的含义。

🙌 根据什么加权求和? Attention is all you need!

- *attention probs*:

	🍏	are	good	for	❤️
🍏	0.4	0.05	0.2	0.05	0.3
are	0.05	0.75	0.1	0.05	0.05
good
for
❤️

$$E_{\text{🍏 in sentence}} = 0.4E_{\text{🍏}} + 0.05 E_{\text{are}} + 0.2 E_{\text{good}} + 0.05 E_{\text{for}} + 0.3 E_{\text{❤️}}$$

👉 Attention Probs

- 如何获取 *attention probs*? 两个词越相关越大?

- 以两个词 *Embedding* 的余弦夹角表示相似度:

$$\text{attention scores}(v_i, v_j) = \cos \langle E_{v_i}, E_{v_j} \rangle$$

- 使用 *Softmax* 归一化:

$$\text{attention probs}(v_i, v_j) = \frac{e^{\text{attention scores}(v_i, v_j)}}{\sum_{k=0}^{\text{sequence length}} e^{\text{attention scores}(v_i, v_k)}}$$

$$E_{v_i \text{ in sentence}} = \sum_{j=0}^{\text{seq len}} \text{attention probs}(v_i, v_j) E_j$$

🤗 如何科学获取 Attention Score?

- *Embedding* 的余弦夹角：与 v_i 相似的特征真的是 v_i 需要的吗？🤔

- 假如句子为：🍏🍏🍏🍏🍏🍏🍏🍏🍏🍏🍏 are good.

- 与 $E_{\text{🍏}}$ 最接近的总是 $E_{\text{🍏}}$, $E_{\text{🍏 in sentence}} = E_{\text{🍏}}$

- *Embedding* \rightarrow (*Query*, *Key*) 🤗🤗🤗

- *Query*: 想查询的特征

- *Key*: 想被查询的特征

$$\text{attention scores}(v_i, v_j) = \cos \langle Q_{v_i}, K_{v_j} \rangle$$

🤔 Attention

- Embedding $\xrightarrow{\text{Linear}}$ (*Query*, *Key*, *Value*)
 $\text{attention scores}(v_i, v_j) = \cos \langle Q_{v_i}, K_{v_j} \rangle$

$$\text{attention probs}(v_i, v_j) = \frac{e^{\text{attention scores}(v_i, v_j)}}{\sum_{k=0}^{\text{sequence length}} e^{\text{attention scores}(v_i, v_k)}}$$

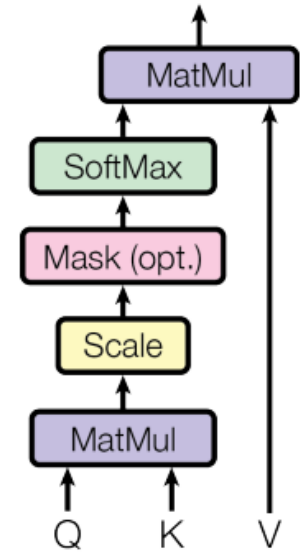
$$V_{v_i \text{ in sentence}} = \sum_{j=0}^{\text{seq len}} \text{attention probs}(v_i, v_j) V_j$$

$$E_{v_i \text{ in sentence}} \xleftarrow{\text{Linear}} V_{v_i \text{ in sentence}}$$

👉 Attention

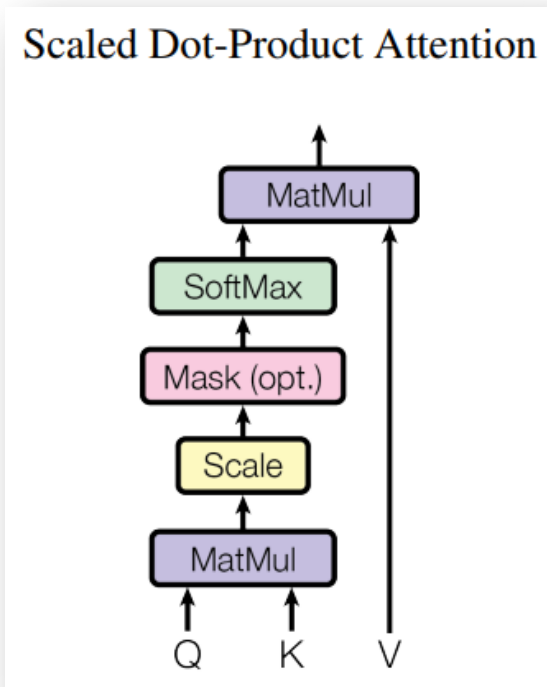
- Input: *Embeddings* $shape = (seq\ len, embedding\ dim)$
- $Q = Q_{proj}(Embeddings)$ $shape = (seq\ len, embedding\ dim)$
- $K = K_{proj}(Embeddings)$ $shape = (seq\ len, embedding\ dim)$
- $V = V_{proj}(Embeddings)$ $shape = (seq\ len, embedding\ dim)$
- $Attention\ Probs(Q, K) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)$ $shape = (seq\ len, seq\ len)$
- $Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$ $shape = (seq\ len, embedding\ dim)$
- $Output = O_{proj}(Attention(Q, K, V))$ $shape = (seq\ len, embedding\ dim)$

Scaled Dot-Product Attention



👉 Attention

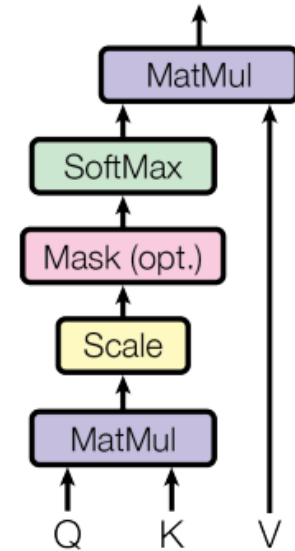
- *Embeddings, Query, Key, Value, Output* 的维度一定都是 *embedding dim* 吗?



👉 Attention

- Input: *Embeddings* $shape = (seq\ len, embedding\ dim)$
- $Q = Q_{proj}(Embeddings)$ $shape = (seq\ len, query\ dim)$
- $K = K_{proj}(Embeddings)$ $shape = (seq\ len, key\ dim), key\ dim = query\ dim$
- $V = V_{proj}(Embeddings)$ $shape = (seq\ len, value\ dim)$
- $Attention\ Probs(Q, K) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)$ $shape = (seq\ len, seq\ len)$
- $Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$ $shape = (seq\ len, value\ dim)$
- $Output = O_{proj}(Attention(Q, K, V))$ $shape = (seq\ len, embedding\ dim)$

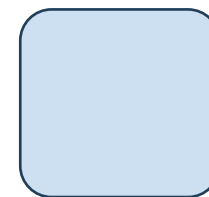
Scaled Dot-Product Attention



👉 Self Attention and Cross Attention

- Self Attention QKV来自同一序列
- Cross Attention Q、KV来自不同序列

Q \ K	🍎	are	good	for	❤️
🍎	0.4	0.05	0.2	0.05	0.3
are	0.05	0.75	0.1	0.05	0.05
good
for
❤️

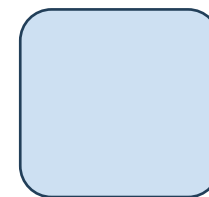


Attention Mask

👉 Self Attention and Cross Attention

- Self Attention QKV来自同一序列
- Cross Attention Q、KV来自不同序列

Q \ K	🍎	are	good	for	❤️
🍎	0.55	0.05	0.4	0	0
are	0.15	0.75	0.1	0	0
good	0	0
for	0	0	0
❤️	0	0	0

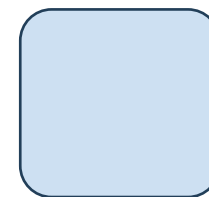


Attention Mask

👉 Self Attention and Cross Attention

- Self Attention QKV来自同一序列
- Cross Attention Q、KV来自不同序列

Q \ K	🍏	are	good	for	❤️
🍏					
are					
good			
for			
❤️			



Attention Mask

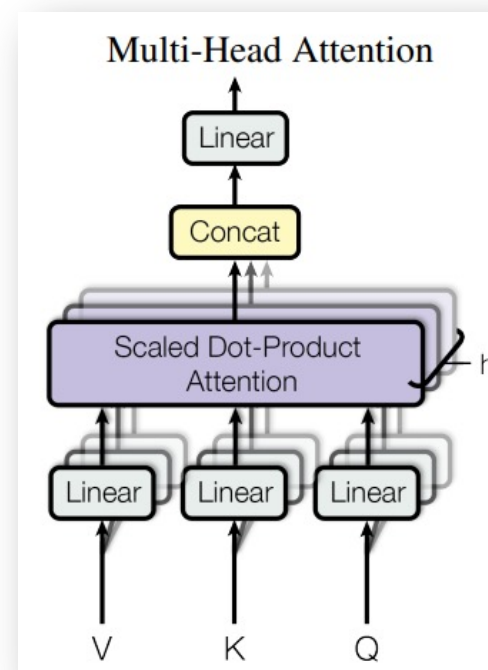
👋 Multi Head Attention 😊😊😊

- $(Q_0, Q_1, Q_2, Q_3) = Q = Q_{proj}(Embeddings)$ $Q_i.shape = (seq\ len, \frac{embeddings\ dim}{num\ head})$
- $(K_0, K_1, K_2, K_3) = K = K_{proj}(Embeddings)$ $K_i.shape = (seq\ len, \frac{embeddings\ dim}{num\ head})$
- $(V_0, V_1, V_2, V_3) = V = V_{proj}(Embeddings)$ $V_i.shape = (seq\ len, \frac{embeddings\ dim}{num\ head})$

- $Output = O_{proj}(Concat(Attention(Q_i, K_i, V_i)))$

- 参数量: $4 \times embeddings\ dim^2$

- 不同Head关注不同的语义关系



🤗 Multi Query Attention ? ? ?

- $(Q_0, Q_1, Q_2, Q_3) = Q = Q_{proj}(Embeddings)$ $Q_i.shape = (seq\ len, \frac{embeddings\ dim}{num\ head})$
- $K = K_{proj}(Embeddings)$ $K.shape = (seq\ len, \frac{embeddings\ dim}{num\ head})$
- $V = V_{proj}(Embeddings)$ $V.shape = (seq\ len, \frac{embeddings\ dim}{num\ head})$
- $Output = O_{proj}(Concat(Attention(Q_i, K, V)))$
- 参数量: $2 \times embeddings\ dim^2 + 2 \times \frac{embeddings\ dim^2}{num\ head}$
- 节省了计算量

🙌 如何加入位置信息 🧭?

- Attention 机制中，并没有考虑向量在 sequence 中的位置。
- *Position embedding*
- 为每一个位置预设一个向量 PE_{pos}

$$E_{v_i \text{ input}} = E_{v_i} + PE_{pos}$$

👉 如何在模型中存储知识🧀?

- 知识：Key-Value Pair: (K , V)

- Neural Memory, 使用 x 查询 k_i :

$$p(k_i|x) \propto e^{k_i \cdot x}$$

$$MN(x) = \sum_{i=1}^{dim} p(k_i|x) \cdot v_i$$
$$K = [k_i], V = [v_i]$$
$$MN(x) = softmax(xK^T)V$$

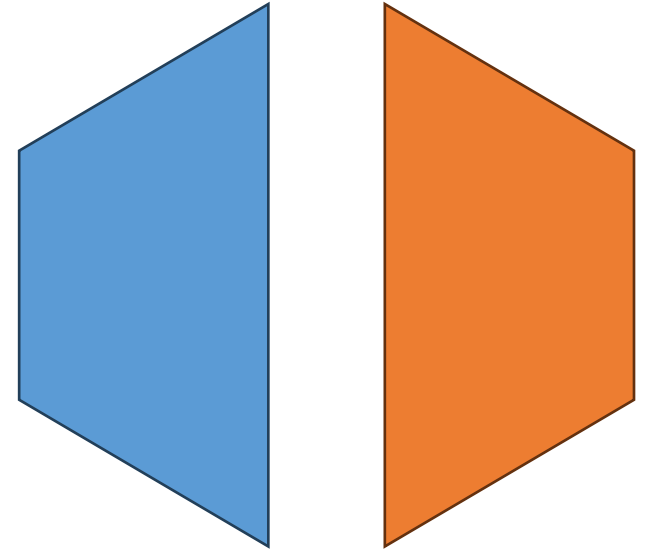
- 用两个线性层实现前馈神经网络:

$$FFN(x) = f(xK^T)V$$

👋 Feedforward Neural Network

$$\text{FFN}(x) = f(xK^T)V$$

- f 一般使用 *relu* 或其变种
- 一般 *inner hidden dim* = 4 *embedding dim*
- 参数量: $8 \times \text{embeddings dim}^2$



🤗 如何基于以上原理构建语言模型？

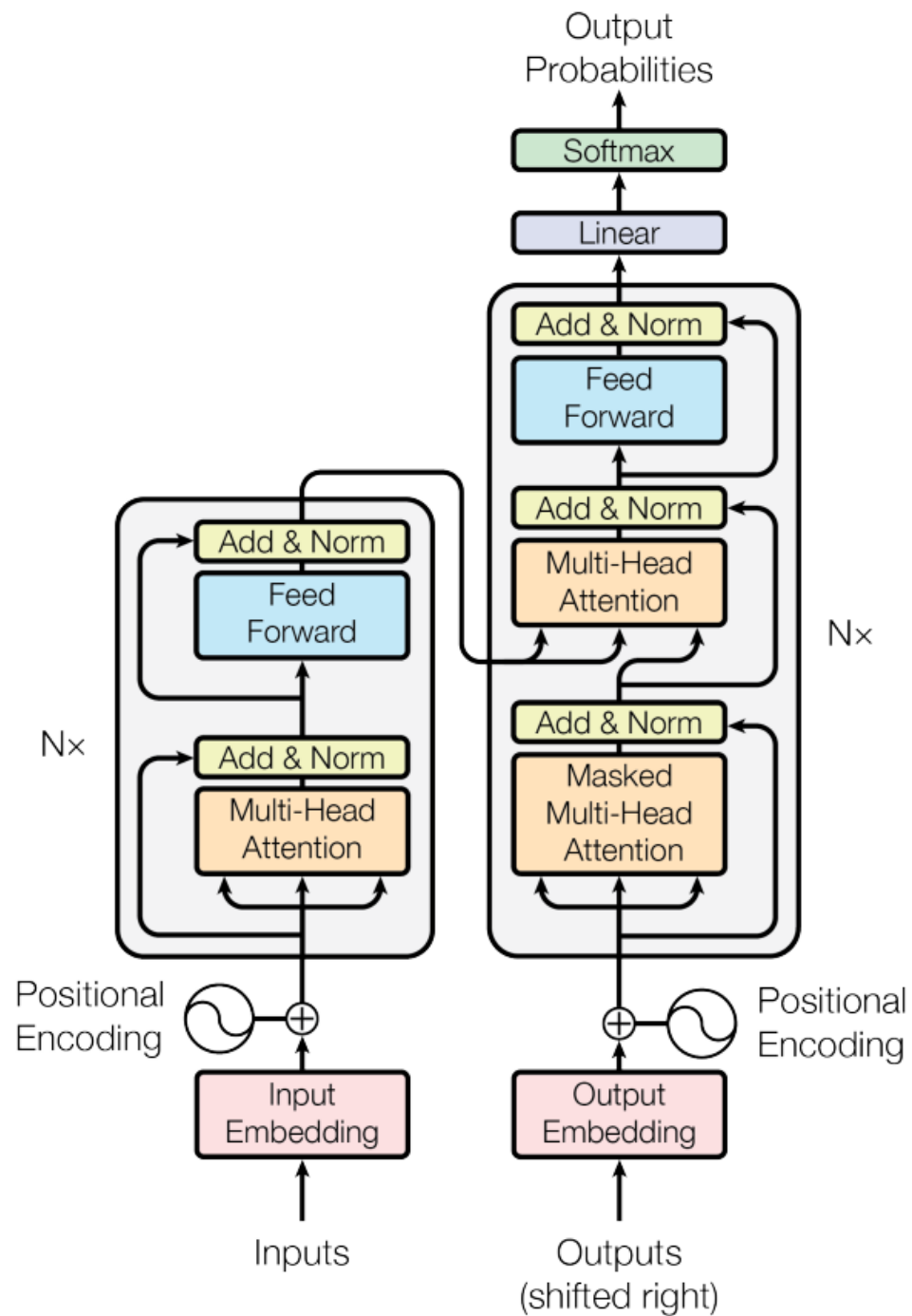
- 语言模型：

$$P(w_i | context)$$

- *context*：上下文
- w_i ：某个位置上的某个词的概率
- 自编码语言模型（如BERT）：
 - “北京在北纬[MASK]度”，求[MASK]填词的概率分布：
 $P(\cdot | \text{北京在北纬[MASK]度})$
- 自编码语言模型（如GPT）：
 - “北京在北”，求下一个字的概率分布： $P(\cdot | \text{北京在北})$

👉 Transformer

- Attention + FFN
- 残差连接：
- ~~$y = F(x)$~~
- $y = x + \Delta x = x + F(x) \checkmark$
- 左侧为Encoder
- 右侧为Decoder



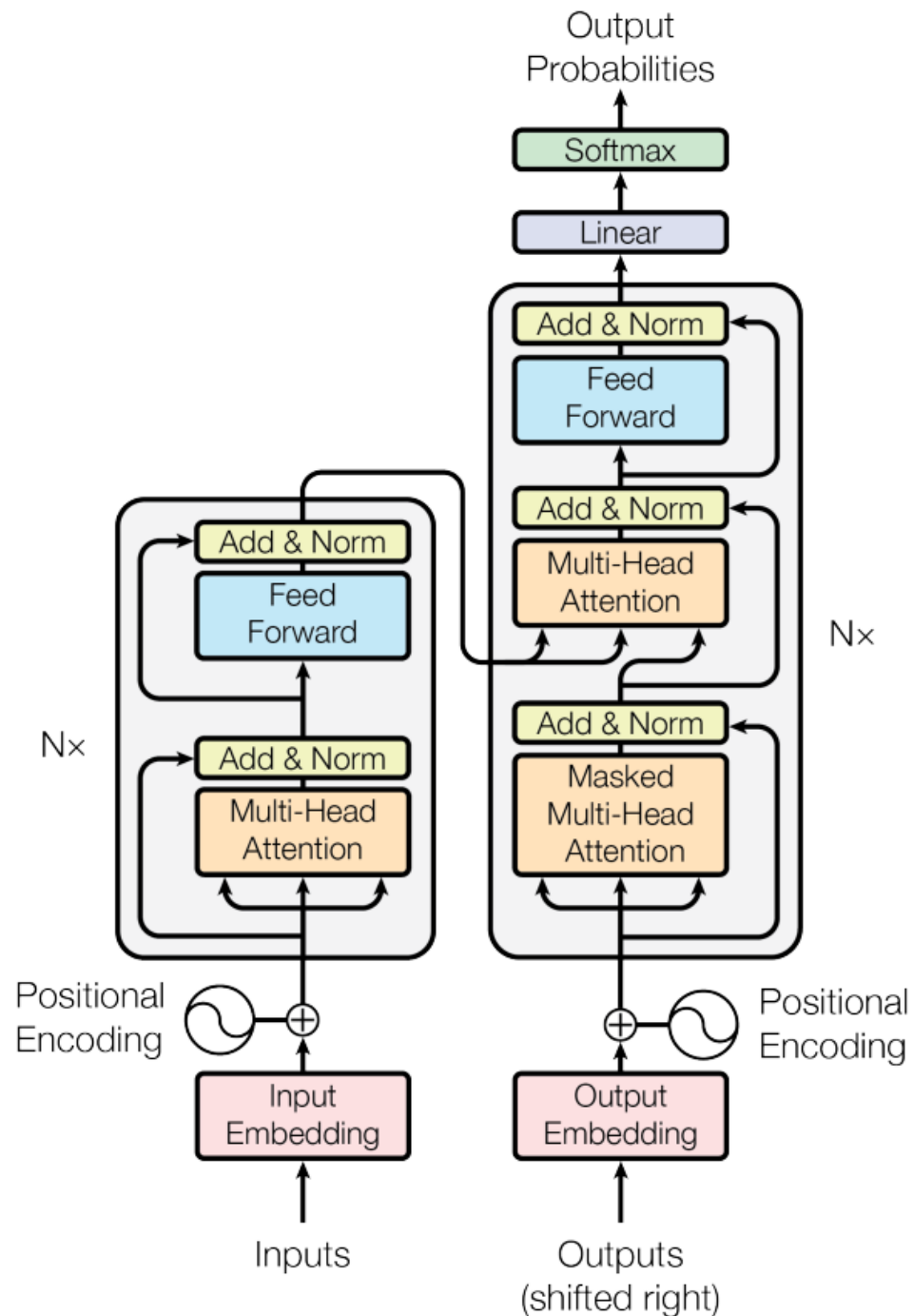
👋 Transformer

- *LM Head*: 一个线性层，输入维度为 *embedding dim*；输出维度为词表大小 *vocab size*。

- 输出每个位置上各词未归一化的对数概率 *logits*:

$$P_{\theta}(\cdot | context) = \text{Softmax}(\text{logits})$$

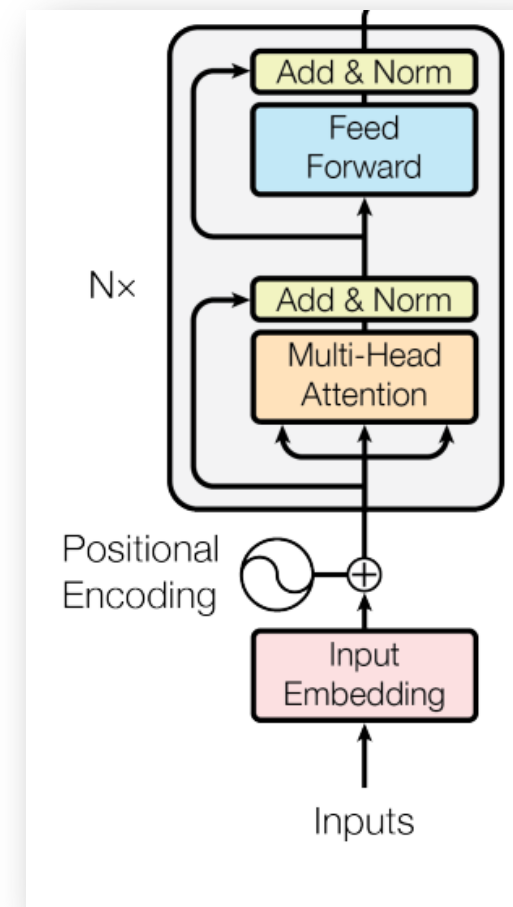
- 左侧为 Encoder
- 右侧为 Decoder



👉 Encoder

- BERT
- 一次前向传播即可计算出所有[MASK]的概率分布
- 文本理解任务

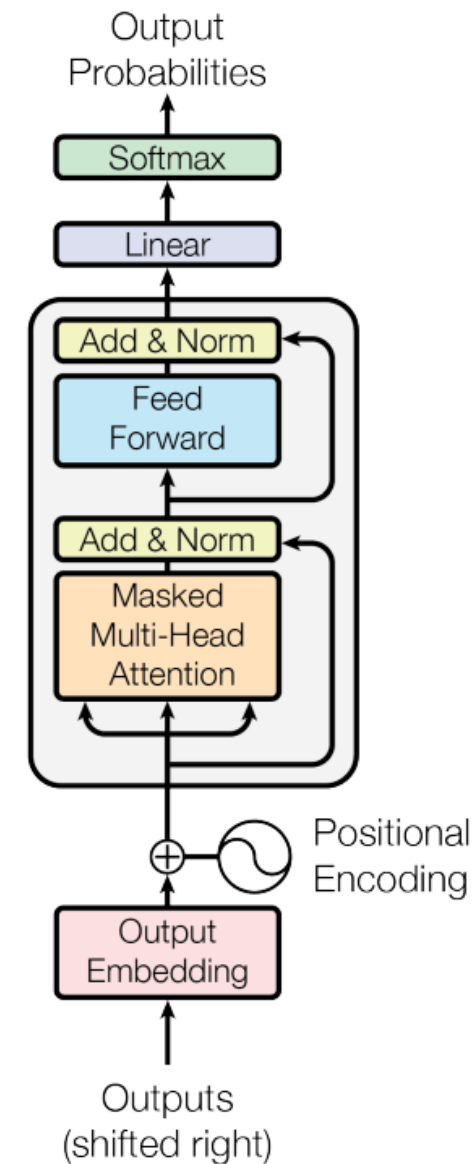
Q \ K	🍎	are	[MASK]	for	❤️
🍎	0.4	0.05	0.2	0.05	0.3
are	0.05	0.75	0.1	0.05	0.05
[MASK]
for
❤️



🤗 Decoder

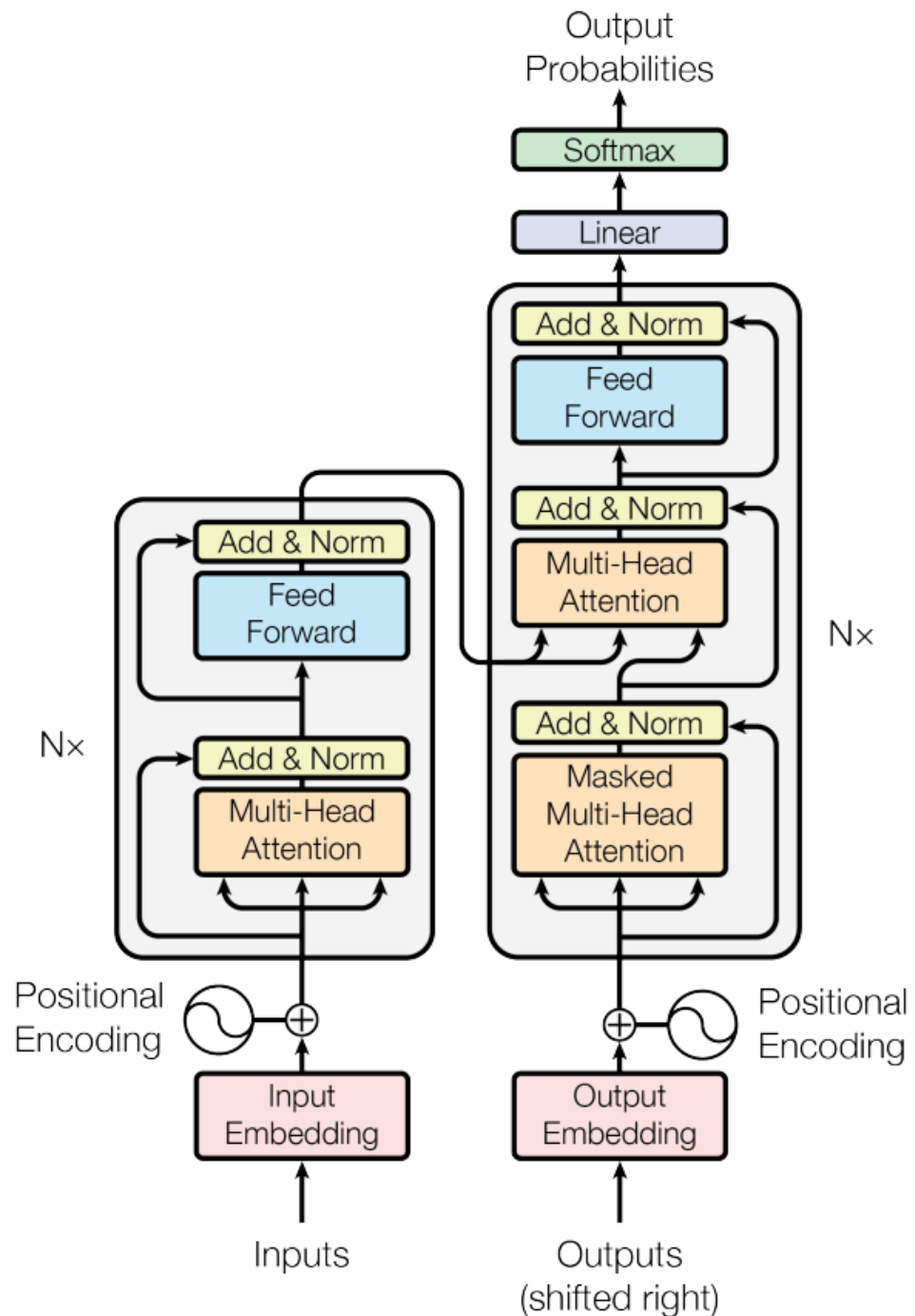
- GPT
- 自回归生成：
- 每次生成下一个词
- Masked Attention：
- 每个词做Attention时只能加权到之前的词
- 文本生成任务

Q \ K	🍎	are	good	for	❤️
🍎	1.0				
are	0.3	0.7			
good		
for	
❤️



👉 Encoder-Decoder

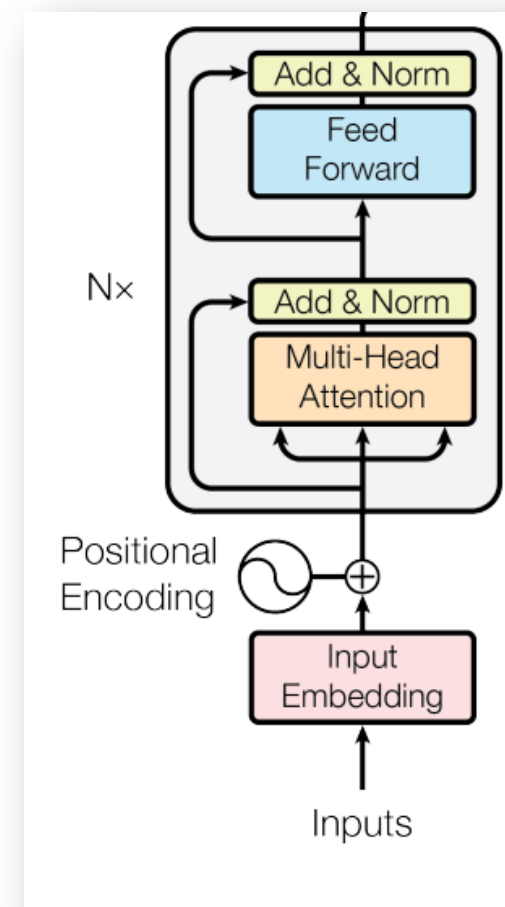
- T5
- Encoder 编码文本
- Decoder 自回归生成
- Decoder 与 Encoder 做 Cross Attention:
 - Decoder Query Encoder 的 Key



😊😊😊 Encoder-Decoder

- GLM

Q \ K	🍏	[MASK]	good	for	❤️
🍏	0.8	0.1	0.1		
[MASK]	0.05	0.35	0.6		
good		
for	
❤️



提问环节

- Prompt:
- 如何表示一个词的含义?
- 如何获取词在句子中的向量表示?
- 如何在模型中存储知识?
- 如何基于以上原理构建语言模型?

语言模型训练

- 如何训练语言模型？
- 如何使用语言模型完成具体下游任务？
- 如何使用有限的硬件资源进行微调？
- 如何让语言模型更好的理解人类意图？

如何训练语言模型?

- 随机初始化权重 θ
- 反复随机初始化权重 θ ，直到权重能较好地完成任务 ❌
- 使用大量人工标注文本对模型进行训练 🤔🤔🤔
- 使用大量无标注文本对模型进行训练 🧐🤔🤔

👉 预训练

- Mask filling or Next token prediction?

- Mask filling:

- 我是[MASK], [MASK]考试没有一次[MASK]。




$$\max_{\theta} (P_{\theta}(\text{大学生}|\text{context}) \cdot P_{\theta}(\text{幼儿园}|\text{context}) \cdot P_{\theta}(\text{参加}|\text{context}))$$

- Next token prediction:

- 我是大学生,

$$\begin{aligned} & \max_{\theta} \left(\prod P_{\theta}(\text{next token}|\text{prefix}) \right) \\ & = \max_{\theta} (P_{\theta}(\text{大学生}|\text{我是}) \cdot P_{\theta}(, |\text{我是大学生})) \end{aligned}$$

如何使用语言模型完成具体下游任务？

- 反复随机初始化权重 θ ，直到权重 θ 能较好地完成任务 
- 在预训练模型的基础上继续使用大量无标注文本训练 
- 使用少量标注数据 (x, y) 模型进行训练 

👉 如何使用语言模型完成具体下游任务？

- 例：分类任务
- 替换预训练模型中的LM Head为Classification Head
- Classification Head输入维度为 *embedding dim*
- 输出维度为类别数目 *num labels*
- ~~然后直接拿去部署~~
- 使用标注的数据 (x, y) 进行训练
 - x : 文本； y : 类别

如何使用有限的硬件资源进行微调？

- LoRA: Low-Rank Adaptation

Low-Rank Adaption

$$\Delta W = W_a W_b$$

$$W = W + W_a W_b$$

$$\text{size}(W_a) = n \times r$$



Fine-Tuning

$$W = W + \Delta W$$

W : New Weights

W : Raw Weights

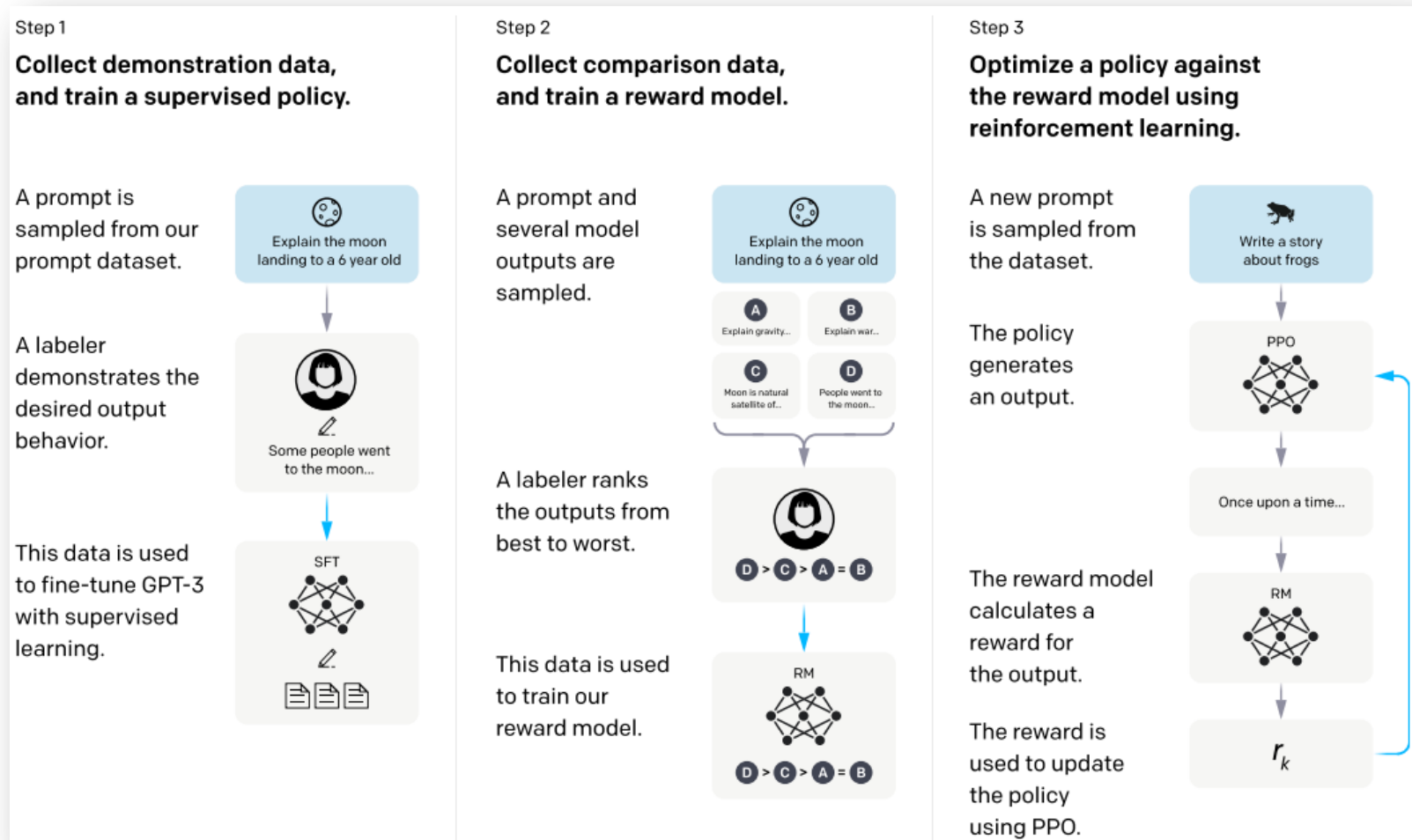
ΔW : Update

$$\text{size}(\Delta W) = n^2$$



Too Big!

👉 如何让语言模型更好的理解人类意图？

- 指令微调
- 使用人工标注的 (prompt, response) 对模型进行微调
- 强化学习



课程实践

- Chat with LM
-  +  problem
- Manual [MASK] filling
- Pretraining
- $\text{Mean}(\text{gpt2}, \text{gpt4}) = \text{gpt3}?$

0. Chat with LM

- 与在线语言模型聊天 (ChatGPT, ChatGLM, Bing, CodeGeeX.....)
- 问一些奇怪的问题：把电池放电到 -1.5v ，再颠倒正负极，电池的电压是多少？
- 分享有趣的结果
- 或者与本地模型交互，如：ChatGLM2-6b

👉0.与本地模型交互

- 配置python环境
- 下载 chatglm2-6b-int4 权重（或者 chatglm2-6b）
- 载入模型
- `model.chat(tokenizer, inputs)`

🤗 1. **A** + **B** Problem

- 下载 bert-base-chinese 权重
- 加载 bert-base-chinese 和 tokenizer
- 通过模型的 Embedding 获取特定词的词向量
- $A+B=?$

🤗 4. Manual [MASK] filling

- 下载 modeling_gpt2.py
- 搜索其中的 [MASK]，并用正确的代码替换

```
class MLP(nn.Module):
    def __init__(self, n_state, config):
        super().__init__()
        nx = config.n_embd # FFN中间维度
        self.c_fc = [MASK](n_state, nx)
        self.c_proj = [MASK](nx, n_state)
        self.act = [MASK] # 激活函数
        self.dropout = nn.Dropout(config.resid_pdrop)

    def forward(self, x):
        h = self.act([MASK])
        h2 = self.c_proj([MASK])
        return self.dropout(h2)
```

```
class MLP(nn.Module):
    def __init__(self, n_state, config):
        super().__init__()
        nx = config.n_embd # FFN中间维度
        self.c_fc = Linear(n_state, nx)
        self.c_proj = Linear(nx, n_state)
        self.act = gelu_new # gpt2-base的激活函数
        self.dropout = nn.Dropout(config.resid_pdrop)

    def forward(self, x):
        h = self.act(self.c_fc(x))
        h2 = self.c_proj(h)
        return self.dropout(h2)
```

👉 8. $\text{Mean}(\text{gpt2}, \text{gpt4}) = \text{gpt3}$?

- 我们将使用 **GPT4** 生成的数据对 **GPT2** 进行 Supervised Fine-Tuning，以此希望 **GPT2** 接近 **GPT3** 的水平。
- 为了减少显存资源占用，我们将在训练过程中使用 **LoRA** 和 **模型量化**。