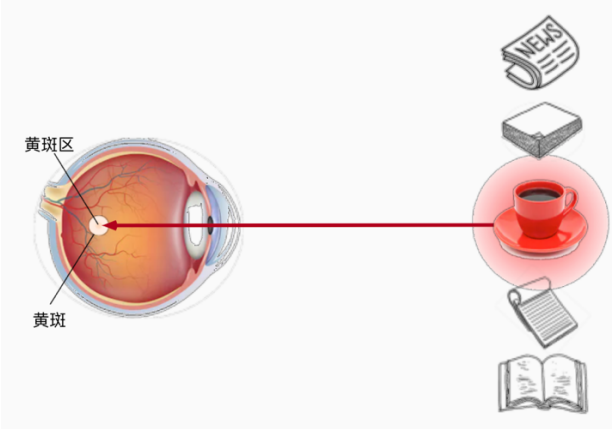
注意力机制

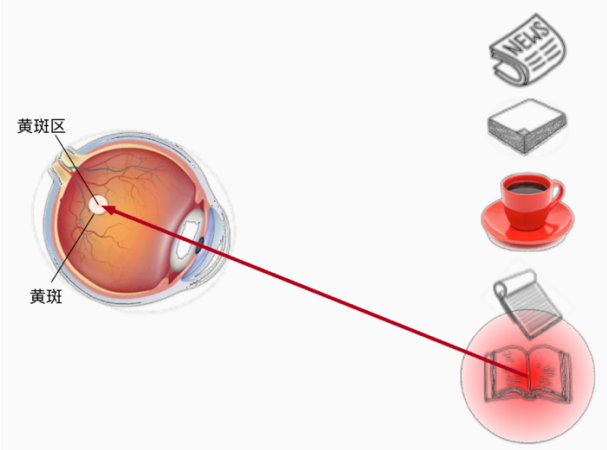
注意力机制

1. 引言 – 心理学

- 动物需要在复杂环境下有效关注值得注意的点
- 心理学框架：人类根据随意线索和不随意线索选择注意点
- **不随意线索**



- **随意线索**



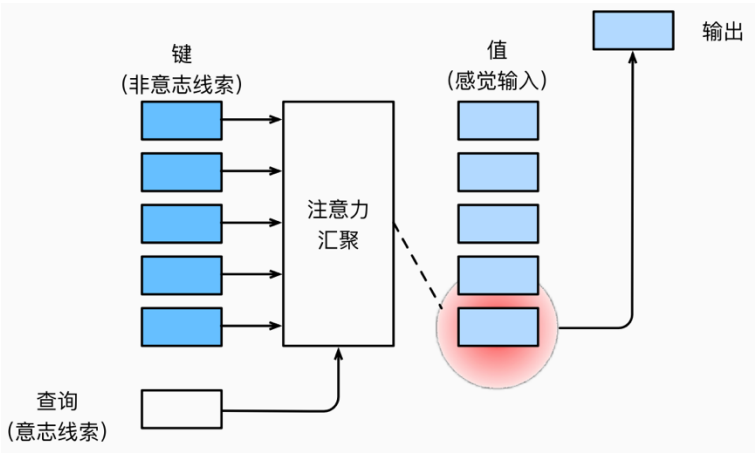
2. 注意力机制

- 卷积，全连接，池化层都只考虑不随意线索
- 注意力机制则显的考虑随意线素

  随意线索被称之为查询 **(query)**

  每个输入是一个值 **(value)** 和不随意线索 **（key）** 的对

  通过注意力池化层来有偏向性的选择选择某些输入



3. 非参注意力池化层

- 给定数据 $(x_i, y_i), i = 1, \ldots, n$
- 平均池化是最简单的方案：$f(x) = \frac{1}{n}\sum_i y_i$
- 更好的方案是60年代提出来的Nadaraya-Watson核回归

$$f(x) = \sum_{i=1}^{n} \frac{K(x - x_i)}{\sum_{j=1}^{n} K(x - x_j)} y_i$$

query　key　value

3.1 Nadaraya-Watson 核回归

- 使用高斯核 $K(u) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{u^2}{2})$
- 那么

$$f(x) = \sum_{i=1}^{n} \frac{\exp\left(-\frac{1}{2}(x - x_i)^2\right)}{\sum_{j=1}^{n} \exp\left(-\frac{1}{2}(x - x_j)^2\right)} y_i$$

$$= \sum_{i=1}^{n} \text{softmax}\left(-\frac{1}{2}(x - x_i)^2\right) y_i$$

4. 参数化的注意力机制

- 在之前基础上引入可以学习的 w

$$f(x) = \sum_{i=1}^{n} \text{softmax}\left(-\frac{1}{2}((x - x_i)w)^2\right) y_i$$

5. 总结

- 心理学认为人通过随意线索和不随意线索选择注意点
- 注意力机制中，通过query（随意线索）和key（不随意线索）来有偏向性的选择输入
  - 可以一般的写作 $f(x) = \sum_i \alpha(x, x_i) y_i$，这里 $\alpha(x, x_i)$ 是注意力权重
  - 早在60年代就有非参数的注意力机制
  - 接下来我们会介绍多个不同的权重设计

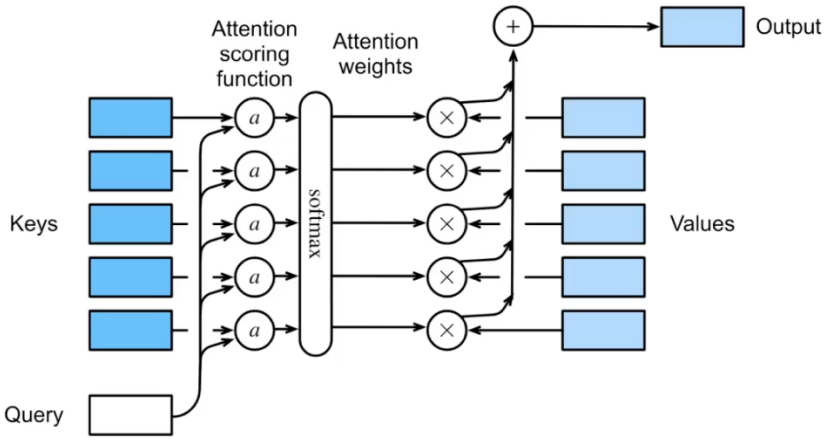
注意力分数

1. 引言

- 回顾

注意力权重　　注意力分数

$$f(x) = \sum_i \alpha(x, x_i) y_i = \sum_{i=1}^{n} \text{softmax}\left(-\frac{1}{2}(x - x_i)^2\right) y_i$$



2. 拓展到高维度

- 假设query $\mathbf{q} \in \mathbb{R}^q$，$m$ 对 key-value $(\mathbf{k}_1, \mathbf{v}_1), \ldots,$ 这里 $\mathbf{k}_i \in \mathbb{R}^k, \mathbf{v}_i \in \mathbb{R}^v$
- 注意力池化层：

$$f(\mathbf{q}, (\mathbf{k}_1, \mathbf{v}_1), \ldots, (\mathbf{k}_m, \mathbf{v}_m)) = \sum_{i=1}^{m} \alpha(\mathbf{q}, \mathbf{k}_i) \mathbf{v}_i \in \mathbb{R}^v$$

$$\alpha(\mathbf{q}, \mathbf{k}_i) = \text{softmax}(a(\mathbf{q}, \mathbf{k}_i)) = \frac{\exp(a(\mathbf{q}, \mathbf{k}_i))}{\sum_{j=1}^{m} \exp(a(\mathbf{q}, \mathbf{k}_j))} \in \mathbb{R}$$

注意力分数

3. Additive Attention

- 可学参数：$\mathbf{W}_k \in \mathbb{R}^{h \times k}, \mathbf{W}_q \in \mathbb{R}^{h \times q}, \mathbf{v} \in \mathbb{R}^h$

$$a(\mathbf{k}, \mathbf{q}) = \mathbf{v}^T \tanh(\mathbf{W}_k \mathbf{k} + \mathbf{W}_q \mathbf{q})$$

- 等价于将 key 和 query **合并**起来后放入到一个**隐藏大小为 h，输出大小为 1** 的**单隐藏层 MLP**

4. Scaled Dot-Product Attention

- 如果query和key都是同样的长度 $\mathbf{q}, \mathbf{k}_i \in \mathbb{R}^d$，那么可以

$$a(\mathbf{q}, \mathbf{k_i}) = \langle \mathbf{q}, \mathbf{k_i} \rangle / \sqrt{d}$$

- 向量化版本
  - $\mathbf{Q} \in \mathbb{R}^{n \times d}$，$\mathbf{K} \in \mathbb{R}^{m \times d}$, $\mathbf{V} \in \mathbb{R}^{m \times v}$
  - 注意力分数：$a(\mathbf{Q}, \mathbf{K}) = \mathbf{Q}\mathbf{K}^T / \sqrt{d} \in \mathbb{R}^{n \times m}$
  - 注意力池化：$f = \text{softmax}\left(a(\mathbf{Q}, \mathbf{K})\right) \mathbf{V} \in \mathbb{R}^{n \times v}$

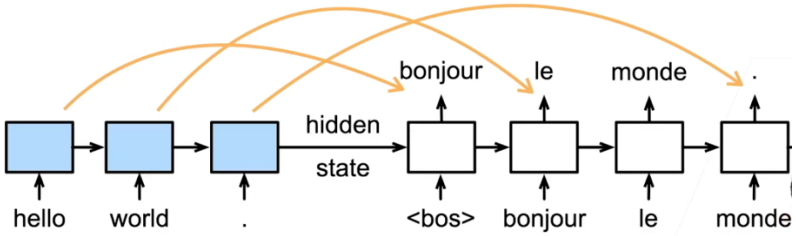
5. 总结

- 注意力分数是query和key的相似度，注意力权重是分数的softmax结果
- 两种常见的分数计算：
  - 将query和key合并起来进入一个单输出单隐藏层的MLP
  - 直接将query和key做内积

使用注意力机制的 seq2seq

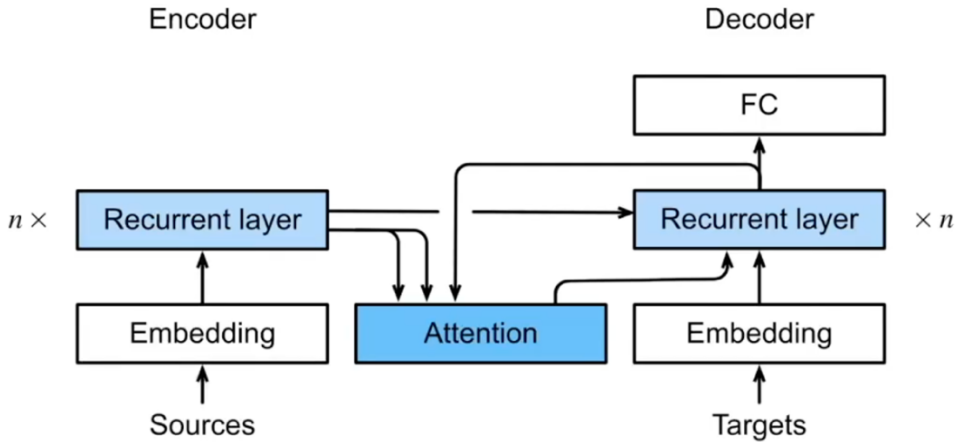
1. 动机

- 机器翻译中，每个生成的词可能相关于源句子中不同的词
- seq2seq 模型中不能对此直接建模



2. 加入注意力

- 编码器对每次词的输出作为 key 和 value（它们是一样的）
- 解码器 RNN 对上一个词的输出是 query



3. 总结

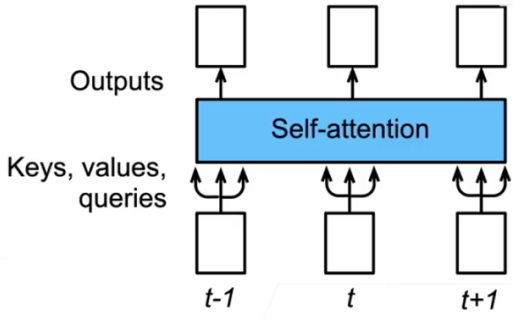
- seq2seq 中通过隐状态在编码器和解码器中传递信息
- 注意力机制可以根据解码器 RNN 的输出来匹配到合适的编码器 RNN 的输出来更有效的传递信息

自注意力

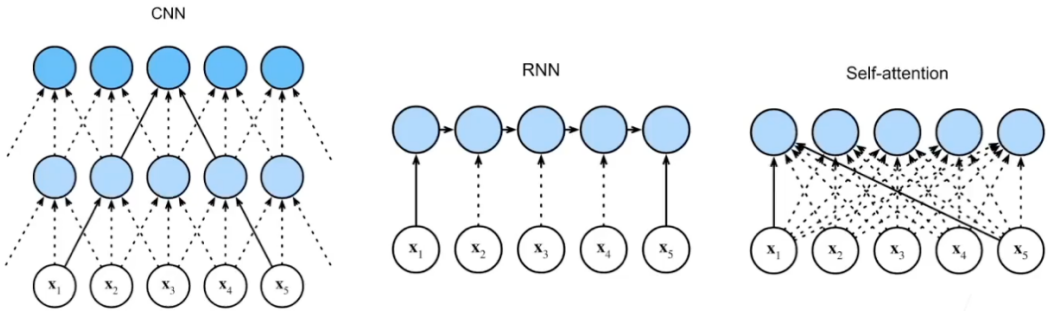
1. 自注意力

- 给定序列 $\mathbf{x}_1, \ldots, \mathbf{x}_n, \forall \mathbf{x}_i \in \mathbb{R}^d$
- 自注意力池化层将 $\mathbf{x}_i$ 当做key，value，query来对序列抽取特征得到 $\mathbf{y}_1, \ldots, \mathbf{y}_n$，这里

$$\mathbf{y}_i = f(\mathbf{x}_i, (\mathbf{x}_1, \mathbf{x}_1), \ldots, (\mathbf{x}_n, \mathbf{x}_n)) \in \mathbb{R}^d$$



1.1 和 CNN，RNN 的对比



	CNN	RNN	自注意力
计算复杂度	O(knd^2)	O(nd^2)	O(n^2d)
并行度	O(n)	O(1)	O(n)
最长路径	O(n/k)	O(n)	O(1)

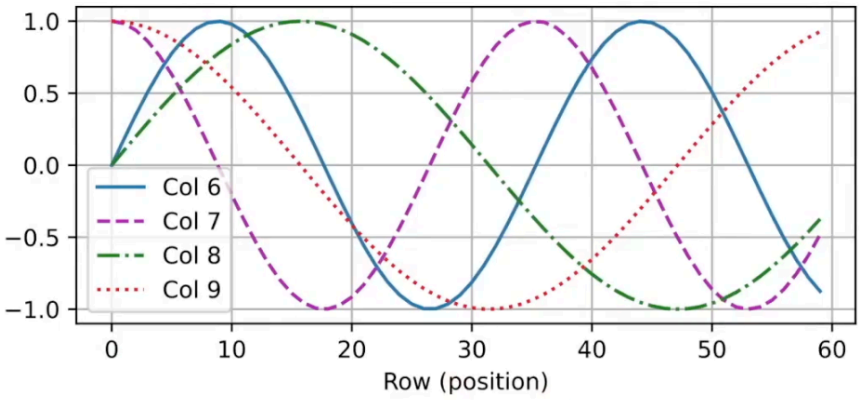
3. 位置编码

- 跟CNN/RNN不同，自注意力并没有记录位置信息
- 位置编码将位置信息注入到输入里
  - 假设长度为 n 的序列是 $\mathbf{X} \in \mathbb{R}^{n \times d}$，那么使用位置编码矩阵 $\mathbf{P} \in \mathbb{R}^{n \times d}$ 来输出 $\mathbf{X} + \mathbf{P}$ 作为自编码输入
- $\mathbf{P}$ 的元素如下计算：

$$p_{i,2j} = \sin\left(\frac{i}{10000^{2j/d}}\right), \quad p_{i,2j+1} = \cos\left(\frac{i}{10000^{2j/d}}\right)$$

3.1 位置编码矩阵

- $\mathbf{P} \in \mathbb{R}^{n \times d}$：$p_{i,2j} = \sin\left(\frac{i}{10000^{2j/d}}\right)$，$p_{i,2j+1} = \cos\left(\frac{i}{10000^{2j/d}}\right)$



3.2 绝对位置信息

- 类似于计算机使用的二进制编码

```
0 in binary is 000
1 in binary is 001
2 in binary is 010
3 in binary is 011
4 in binary is 100
5 in binary is 101
6 in binary is 110
7 in binary is 111
```

3.3 相对位置信息

- 位置于 $i+\delta$ 处的位置编码可以线性投影位置 i 处的位置编码来表示
- 记 $\omega_j = 1/10000^{2j/d}$，那么

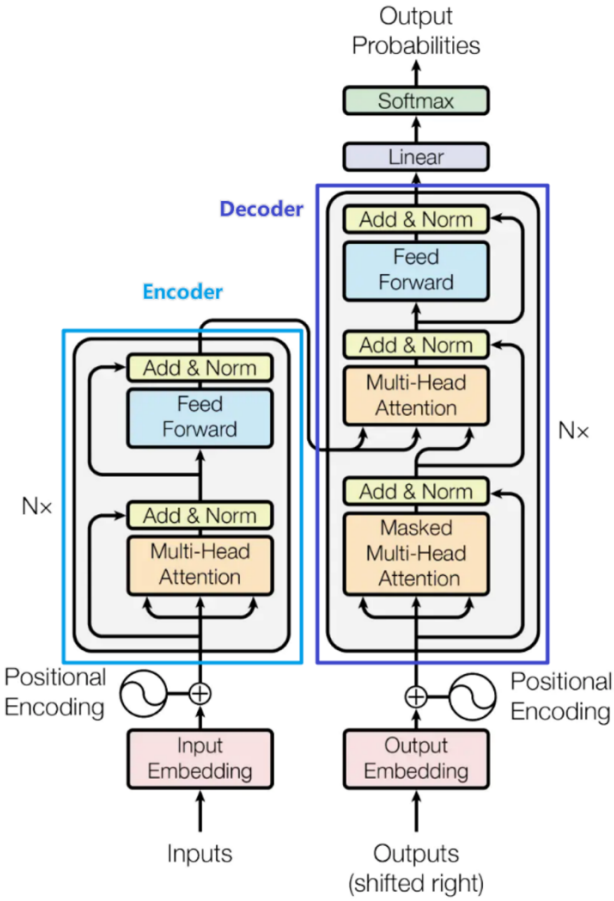
$$\begin{bmatrix} \cos(\delta\omega_j) & \sin(\delta\omega_j) \\ -\sin(\delta\omega_j) & \cos(\delta\omega_j) \end{bmatrix} \begin{bmatrix} p_{i,2j} \\ p_{i,2j+1} \end{bmatrix} = \begin{bmatrix} p_{i+\delta,2j} \\ p_{i+\delta,2j+1} \end{bmatrix}$$

投影矩阵跟 i 无关

Transformer

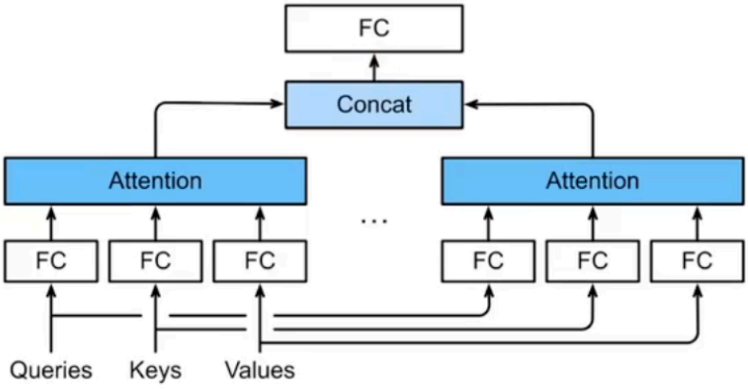
1. Transformer 架构

- 基于编码器-解码器架构来处理序列对
- 跟使用注意力的 seq2seq 不同，Transformer 是纯基于 Self-Attention



2. 多头注意力

- 对同一 key，value，query，希望抽取不同的信息
  例如短距离关系和长距离关系
- 多头注意力使用 h 个独立的注意力池化
  合并各个头(head) 输出得到最终输出



- query $\mathbf{q} \in \mathbb{R}^{d_q}$，key $\mathbf{k} \in \mathbb{R}^{d_k}$，value $\mathbf{v} \in \mathbb{R}^{d_v}$
- 头 i 的可学习参数 $\mathbf{W}_i^{(q)} \in \mathbb{R}^{p_q \times d_q}$, $\mathbf{W}_i^{(k)} \in \mathbb{R}^{p_k \times d_k}$, $\mathbf{W}_i^{(v)} \in \mathbb{R}^{p_v \times d_v}$
- 头 i 的输出 $\mathbf{h}_i = f(\mathbf{W}_i^{(q)}\mathbf{q}, \mathbf{W}_i^{(k)}\mathbf{k}, \mathbf{W}_i^{(v)}\mathbf{v}) \in \mathbb{R}^{p_v}$
- 输出的可学习参数 $\mathbf{W}_o \in \mathbb{R}^{p_o \times hp_v}$
- 多头注意力的输出

$$\mathbf{W}_o \begin{bmatrix} \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_h \end{bmatrix} \in \mathbb{R}^{p_o}$$

2.1 带掩码的多头注意力

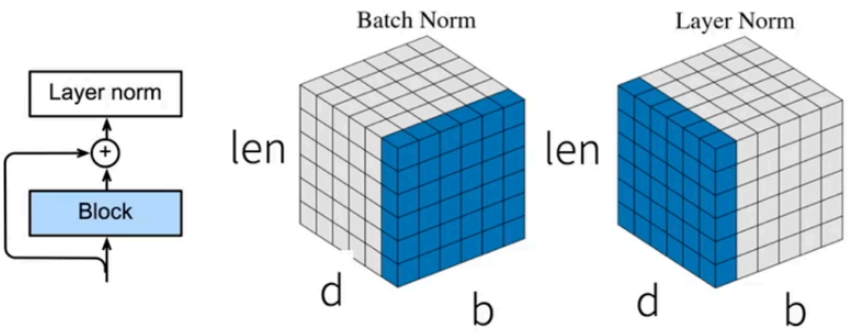
- 解码器对序列中一个元素输出时，不应该考虑该元素之后的元素
- 可以通过掩码来实现
  - 也就是计算 $\mathbf{x}_i$ 输出时，假装当前序列长度为 i

3. 基于位置的前馈网络

- 将输入形状由 (b, n, d) 变换成 (bn, d)
- 作用两个全连接层
- 输出形状由 (bn, d) 变化回 (b, n, d)
- 等价于两层核窗口为1的一维卷积层

4. 层归一化

- 批量归一化对每个特征/通道里元素进行归一化
  - 不适合序列长度会变的NLP应用
- 层归一化对每个样本里的元素进行归一化



5. 信息传递

- 编码器中的输出 $\mathbf{y}_1, \ldots, \mathbf{y}_n$
- 将其作为解码中第 i 个Transformer块中多头注意力的key和value
  - 它的query来自目标序列
- 意味着编码器和解码器中块的个数和输出维度都是一样的

6. 预测

- 预测第 $t+1$ 个输出时
- 解码器中输入前 t 个预测值
  - 在自注意力中，前 t 个预测值作为key和value，第 t 个预测值还作为query