Beijing Forest Studio

北京理工大学信息系统及安全对抗实验中心

注意力机制

Attention Mechanism

白崇有　硕士研究生

2018年10月07日

内容提要

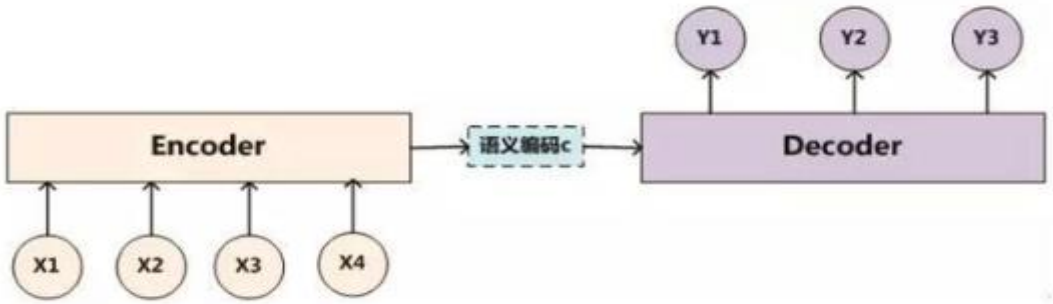
- 背景简介
- 基本概念
- 算法原理
- 优劣分析
- 应用总结
- 参考文献

背景简介

- 预期收获
  - 1. 了解Encoder-Decoder模型
  - 2. 理解注意力机制的算法原理
  - 3. 了解注意力机制在自然语言处理中的应用

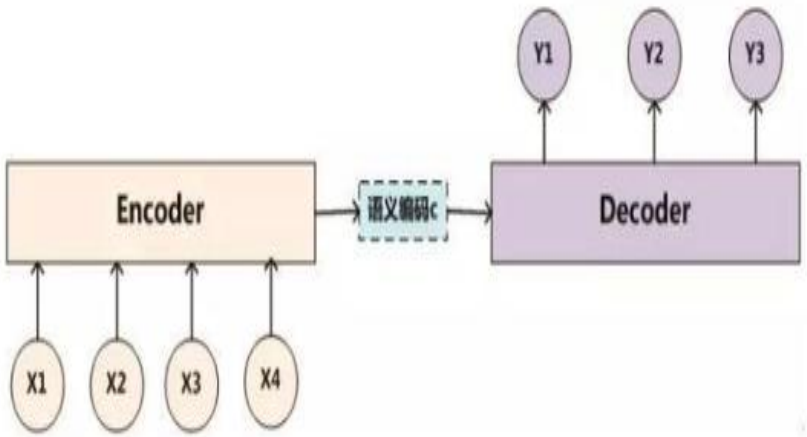
背景简介

- 何为Encoder-Decoder模型?
  - Encoder（编码器）是将源序列（X1,X2,X3,X4,...）转化为一个固定长度的上下文向量C；Decoder（解码器）是将上下文向量C转化为目标序列（ Y1,Y2,Y3,Y4,... ）；可以把它看作适合处理由一个句子（或篇章）生成另外一个句子（或篇章）的通用处理模型。
  - Encoder最后一个时间步的状态作为整个句子的中间语义上下文向量C，上下文向量直接作为Decoder的输入。



背景简介

- 目标序列的生成过程
  - 假设Decoder部分采用的是LSTM单元；



$$Y_i = f(c, Y_{i-1}, s_{i-1})$$

- 可以看出，在生成目标序列时，不论生成哪个单词,Decoder使用的是相同的上下文向量c，这意味着输入句子中任意单词对生成某个目标单词Y_i来说影响力都是相同的。

背景简介

- 以机器翻译为例
  - 输入英文句子
    - Tom chase Jerry
  - 输出中文序列
    - “汤姆”，“追逐”，“杰瑞”
- 比如，Decoder在输出“杰瑞”这个中文单词的时候，输入的每个英文单词对于输出目标单词“杰瑞”的贡献是相同的，很明显这里是不太合理的，显然“Jerry”对于翻译成“杰瑞”更重要，但是该模型是无法体现这一点的。

背景简介

- 算法提出的原因
  - 最基本的Encoder-Decoder模型存在一个问题：不论输入句子的长短,Encoder都将最后一个状态作为Decoder的输入（可能作为初始化，也可能作为每一时刻的输入）；Encoder的状态有限，存储不了太多的信息，如果输入句子比较长，会丢失很多细节信息，这使模型对于长输入序列的学习效果很差（解码效果很差）。
- 提出问题
  - 为了使Decoder在每一时刻的输入都有所不同，使处理长输入句子时效果更好，在本次演讲中我们引入了注意力机制。

基本概念

- 基本概念
  - Query:在Encoder-Decoder模型中Decoder某一时刻的状态称作Query;
  - Key/Value :在Encoder-Decoder模型中Encoder的输出称作Key/Value，在自然语言处理中，Key/Value常常是相同的;

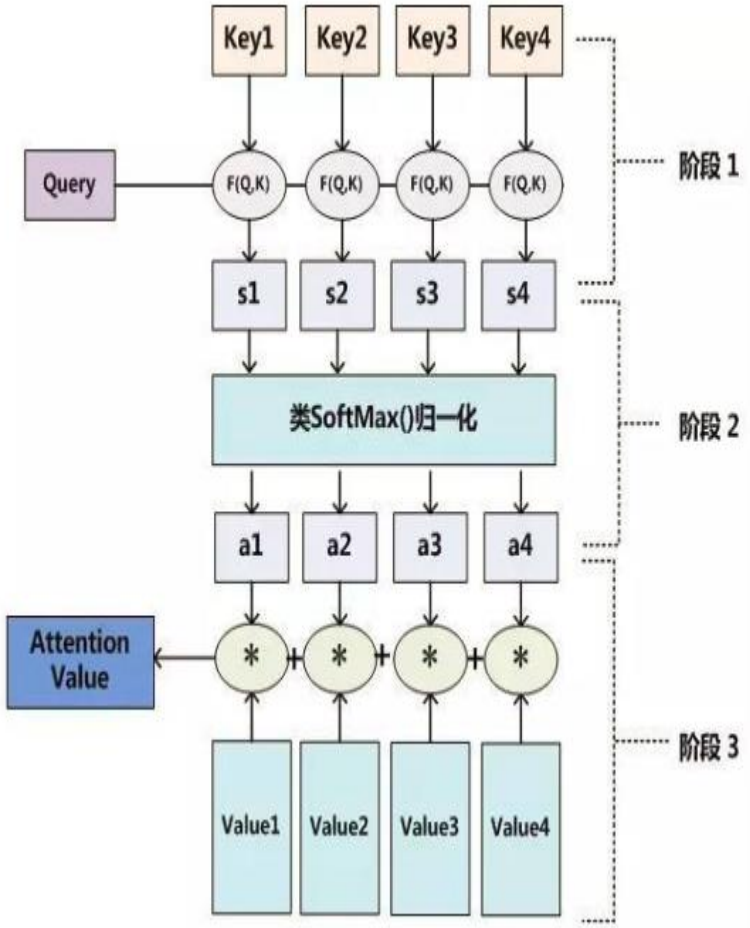
算法原理

T	基于Query Key和Value，求得不同时刻的上下文向量
I	Query Key Value
P	1. 相似度计算 2. 权值归一化 3. 加权求和
O	Attention向量

P	为了使Decoder根据时刻的不同，让每一时刻的输入都有所不同，同时，为了使处理长输入句子时效果更好
C	明确Query Key Value
D	相似度的计算
L	CCF A类会议

算法原理

- Attention机制的计算过



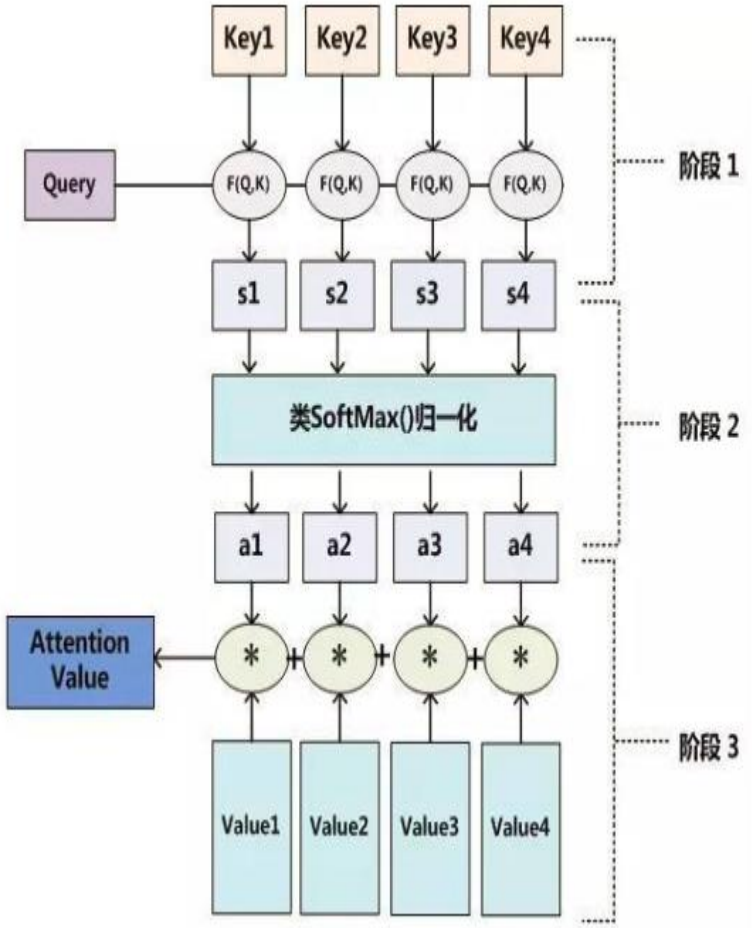
- 第一阶段：将Query和每个Key进行相似度计算得到权重，常用的相似度函数有点积，拼接，感知机等。

算法原理

- 相似度函数
  - $F(Q,K_i) = Q^T K_i$ dot
  - $F(Q,K_i) = Q^T w_a K_i$ general
  - $F(Q,K_i) = w_a [Q; K_i]$ concat
  - $\mathrm{F(Q,K_i)} = v_a^T \tanh(W_a Q + U_a K_i)$ perceptron
  - ……

算法原理

- Attention机制的计算过

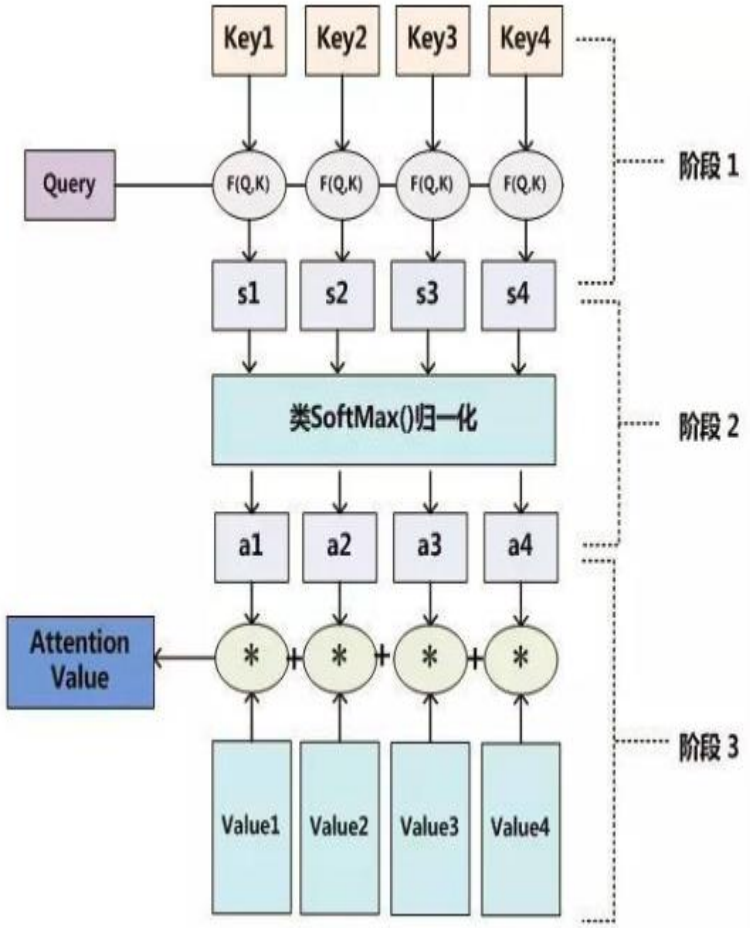


- 第二阶段：一般是使用一个Softmax函数对第一阶段求得的权重进行归一化。
- $a_i = softmax\big(F(Q, K_i)\big)$

$$= \frac{\exp(F(Q, K_i))}{\sum_j \exp(F(Q, K_j))}$$

算法原理

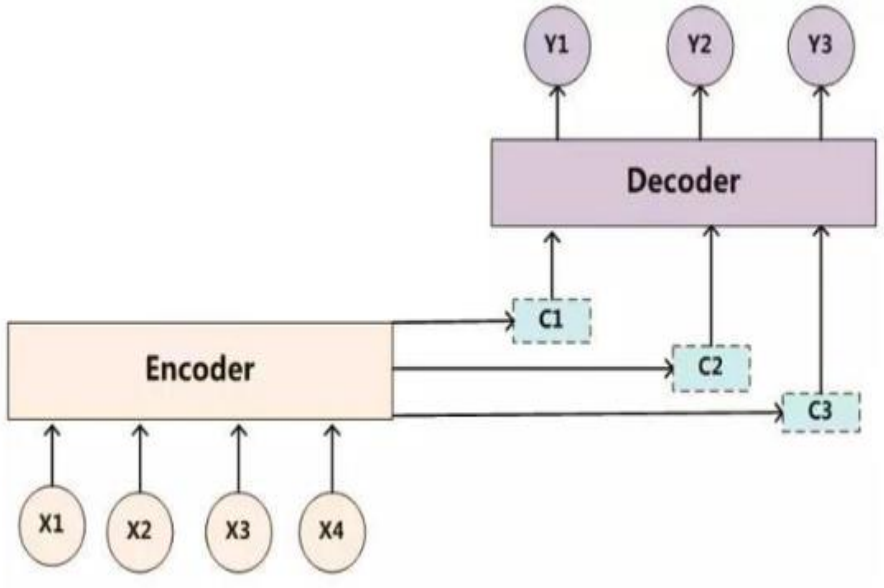
- Attention机制的计算过



- 第三阶段：第二阶段的计算结果a_i即为$Value_i$对应的权重系数，然后每个权重系数和相应的键值value进行加权求和得到最后attention值。
- $Attention(Query, Key, Value) = \sum_i a_i V_i$

算法原理

- 增加了注意力机制的Encoder-Decoder模型
  - 不同点：在生成每个单词的时候，原先都是相同的中间上下文向量C会被替换成根据当前输出单词来调整成加入注意力机制的不断变化的C。



$$Y_i = f(c_i, s_{i-1}, Y_{i-1})$$

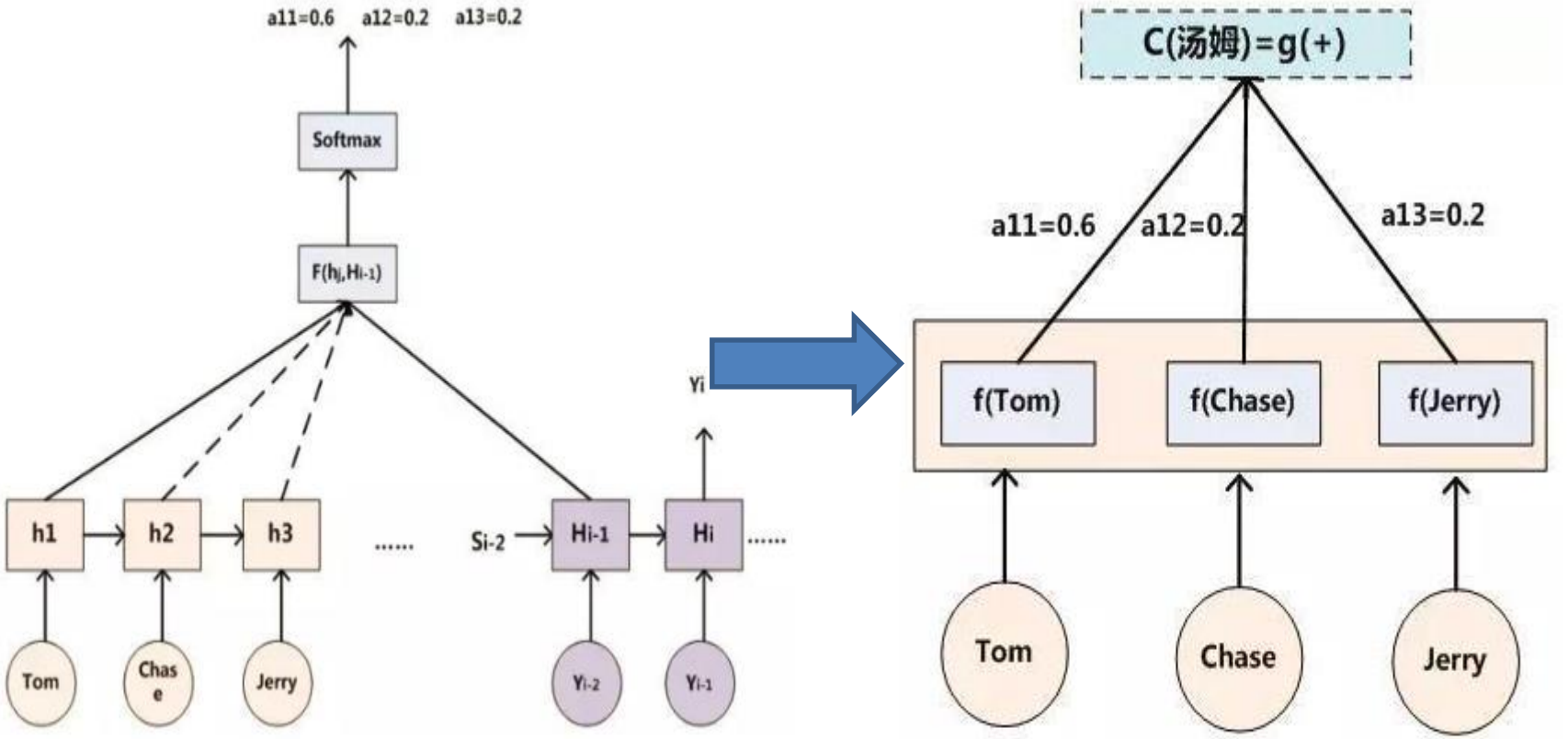
算法原理

- 以机器翻译为例解释如何通过attention机制产生C
  - 输入英文句子
    - Tom chase Jerry
  - 输出中文序列
    - “汤姆”，“追逐”，“杰瑞”

- 翻译中文单词“汤姆”的时候，如何形成中间上下文向量表示$C_{(汤姆)}$呢?

算法原理

- 以机器翻译为例解释如何通过attention机制产生C(汤姆)



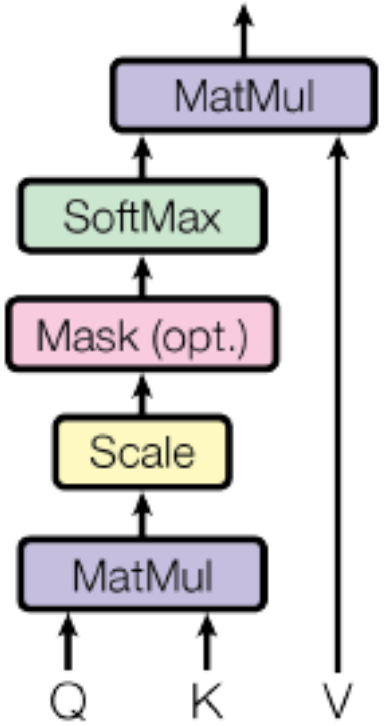
算法原理

- Self Attention
  - 和Attention的相同点
    - Self Attention和Attention的具体计算过程是一样的；
  - 和Attention的不同点
    - 1. Q≠ K ≠V 或者 Q≠ K = V ⟷ Attention

      Q=K=V ⟷ Self Attention；
    - 2. Attention是目标序列和输入序列之间的一种单词对齐机制，Self Attention可以捕获句子中长距离的相互依赖的特征。

算法原理

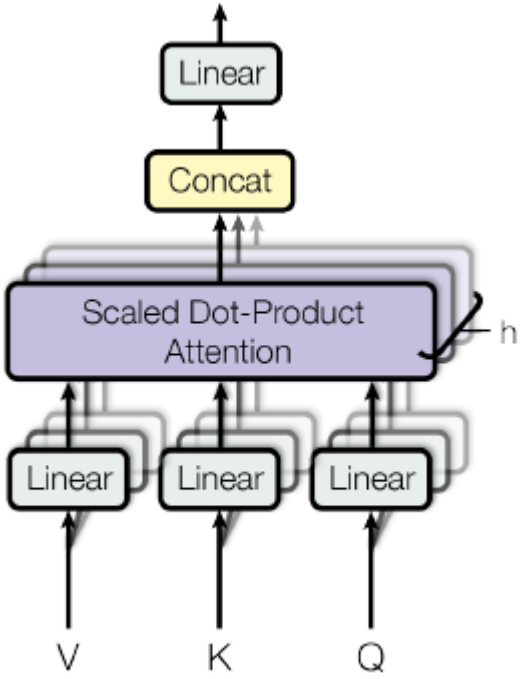
- scaled dot-Product(放缩点积) attention



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

算法原理

- Multi-head attention



$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

优劣分析

- 优势
  - 1.打破了传统编码器-解码器结构在编解码时都依赖于内部一个固定长度向量的限制；
  - 2.相比于RNN，self attention和CNN一样不依赖于前一时刻的计算，可以很好的并行；
  - 3.self attention 能够一步到位捕捉到全局的联系，因为它直接把序列两两比较，而RNN要逐步递归才能获得全局信息，CNN 事实上只能获取局部信息，但可以通过层叠来增大感受野。
- 劣势
  - 1.无法对位置信息进行很好地建模；
  - 2.使用attention机制之后会增加计算量。

应用总结

- 算法的应用领域
  - 机器翻译
  - 阅读理解
  - 语义角色标注
  - 关系抽取等

参考文献

[1]Vaswani A, Shazeer N, Parmar N, et al. Attention Is All You Need[J]. 2017.
[2]Tan Z, Wang M, Xie J, et al. Deep Semantic Role Labeling with Self-Attention. AAAI 2018.
[3]Verga P, Strubell E, Mccallum A. Simultaneously Self-Attending to All Mentions for Full-Abstract Biological Relation Extraction[J]. 2018.
[4]https://baijiahao.baidu.com/s?id=1587926245504773589&wfr=spider&for=pc

谢谢！

大成若缺，其用不弊。大盈若冲，其用不穷。大直若屈。大巧若拙。大辩若讷。静胜躁，寒胜热。清静为天下正。