Report 1

> 2023.2.20-2023.2.28

A Neural Attention Model for Abstractive Sentence Summarization

- paper

- Attention-based model that generates abstractive summarization

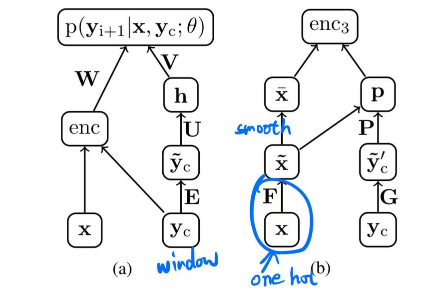


Figure 3: (a) A network diagram for the NNLM decoder with additional encoder element. (b) A network diagram for the attention-based encoder enc_3.

- decoder: NNLM, encoder: attention-based, beam search to generate summarization

- NLL(θ) = $-\sum_{j=1}^{J}$log$p(y^j|x^j;\theta)$ = $-\sum_{j=1}^{J}\sum_{i=1}^{N-1}logp(y^j_{i+1}|x^j, y_c;\theta)$

- 本文将 seq2seq用于文本摘要技术。具体来说就是利用上图模型算出$p(y_{i+1}|y_c, x;\theta)$,然后用机器学习将NLL最大化。生成summary是采取beam search，即每次遍历整个词典找概率最大的k个词形成summary。此方法和其他模型比可以准确找到关键词，但词的正确顺序难以保证。代码不是用python写的，没看懂。

Attention Is All You Need

- paper & code （代码实现下周看）

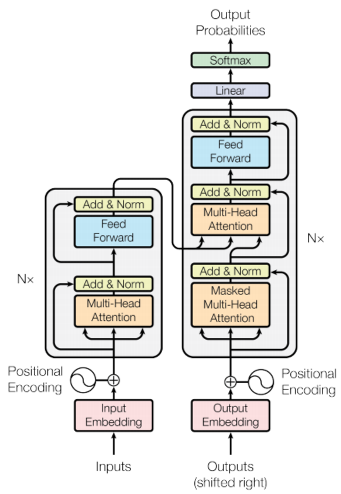


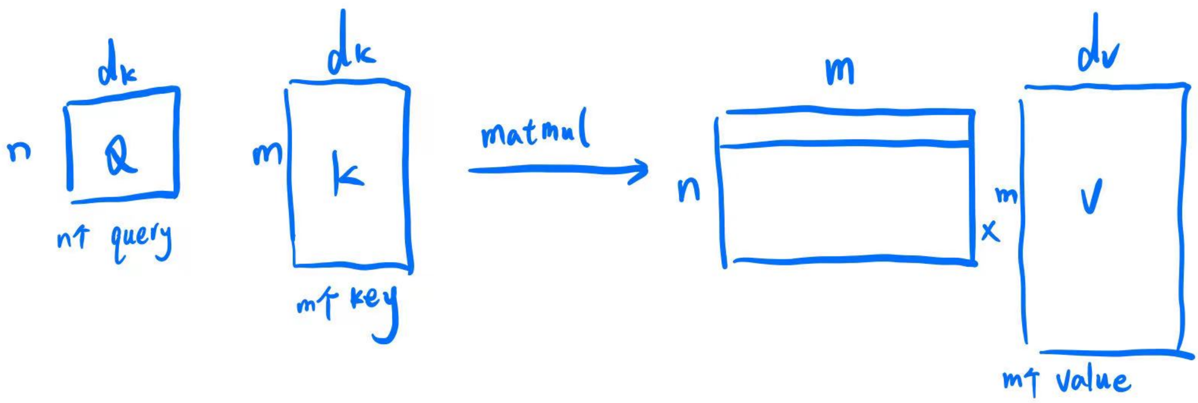
Figure 1: The Transformer - model architecture.

> 输入：n个长为d的向量。第一个注意力层三个输入：key, value, query，自注意力机制。第二个注意力层类似。第三个key, value来自encoder，query来自decoder上一个输出

Attention

- Scaled Dot-Product Attention

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

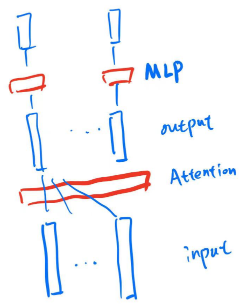


- Multi-Head attention

  MultiHead(Q, K, V) = Concat(head$_1$, ..., head$_n$)W^O

  where head$_i$ = Attention(QW_i^Q, KW_i^K, VW_i^V)

  本质上是给h次机会将向量投影，投影矩阵W是用来学习的，下面三个维度均为$\mathbb{R}^{d_{model}\times d_k}$，上面为$\mathbb{R}^{hd_v\times d_{model}}$，$d_k$ = d_v = d_{model}/h



- Transformer大致结构

Coding

```python
import torch.nn as nn

class Model(nn.Module):
    def __init__(self):
        super().__init__()


    def forward(self,input):
        output=input+1
        print(output)

model = Model()          #实例化
input = torch.tensor(1) #输入为1
model(input)             #输出为2
```

- super().**init**()调用父类的init
- nn.Module的forwar函数在实例化的时候不需要被调用，即不需要model.forward(input)