2017
应用数学

论文阅读报告
R-Net

By 孙付
17210240013
January 10, 2018

1 问题描述

SQuAD 数据集是斯坦福大学发布的公开数据集，提供 train 和 dev 训练集，未公开 test 数据集。该数据集上的数据给定一段 passage 和一个 Question，求取 Answer，Answer 为文中出现的一个东西，Passage、Question 和 Answer 对如下所示。

> Passage: Tesla later approached Morgan to ask for more funds to build a more powerful transmitter. When asked where all the money had gone, Tesla responded by saying that he was affected by the Panic of 1901, which he (Morgan) had caused. Morgan was shocked by the reminder of his part in the stock market crash and by Tesla' s breach of contract by asking for more funds. Tesla wrote another plea to Morgan, but it was also fruitless. Morgan still owed Tesla money on the original agreement, and Tesla had been facing foreclosure even before construction of the tower began.
> Question: On what did Tesla blame for the loss of the initial money?
> Answer: Panic of 1901

2 模型结构

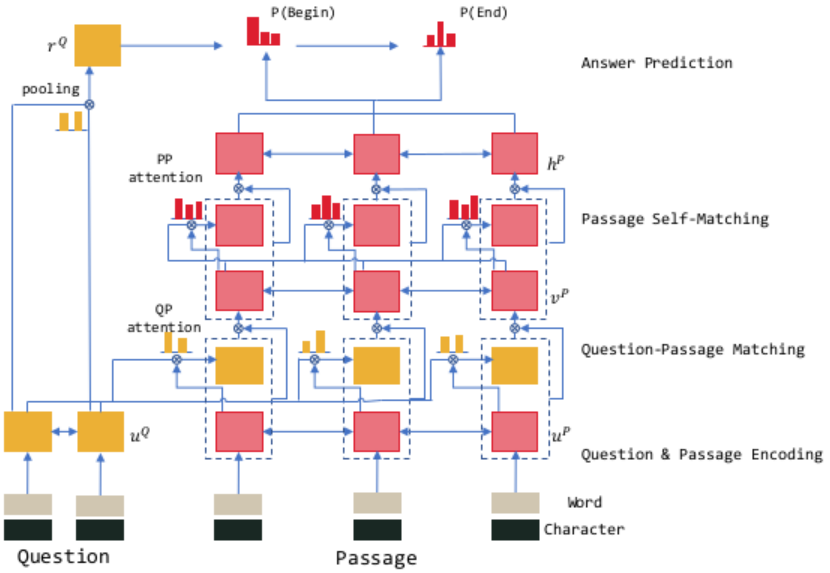

Figure 1: R-Net

模型看起来很复杂，其实主要就是对 attention 的运用，而且对 attention 运用了非常多的次数，个人感觉此结构对给定的 passage 和 question 进行了非常深入的挖掘，但是仅仅是对文本本身的挖掘，和语义理解没有什么关系。由于文中用到了 attention 来对 passage 和 question（passage）进行 match，最后用 pointer networks 来选择答案，所以下面对前面的文章由远到近的讲一下。

3 Neural Machine Translation by Jointly Learning to Align and Translate

原论文是用于解决机器翻译的问题，结构如下 简单推导（本文主题不在此，简单介绍），定义条件概率：

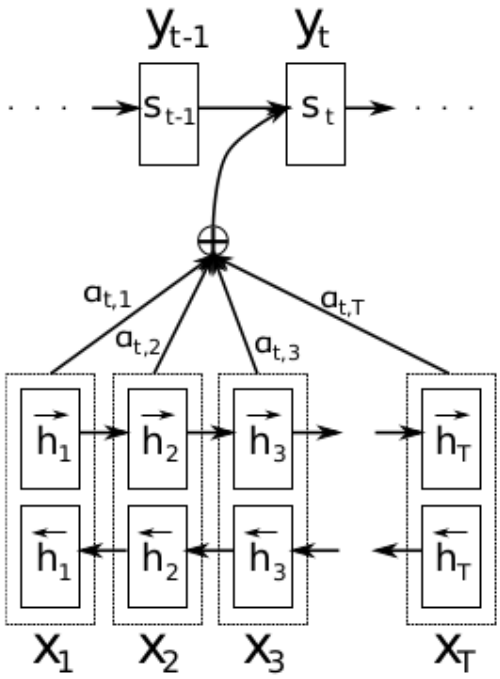

Figure 2: Attention

$$p(y_i|y1,...,y_{i-1},x) = g(y_{i-1},s_i,c_i)$$

其中 s_i 是 RNN 的 hidden state，y 是翻译的输出，x 是输入，s_i 通过

$$s_i = f(s_{i-1},y_{i-1},c_i)$$

计算，向量 c_i 由处理输入的 BiRNN 的隐藏状态决定，

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

，权重 α_{ij} 通过

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ij})}$$

计算且 $e_{ij} = a(s_{i-1},h_j)$，这个函数用于对第 j 个位置的输入和第 i 个位置的输出打分，看看他们之间有多 match，原文是用一个神经网络 train 的，后面的用于解决 QA 问题的一般都是直接计算的，用一个矩阵表示。如在论文 Text Understanding with the Attention Sum Reader Network 中的方法：

- We compute a vector embedding of the query.

- We compute a vector embedding of each indi-vidual word in the context of the whole doc-ument (contextual embedding).
- Using a dot product between the question embedding and the contextual embedding of each occurrence of a candidate answer in the document, we select the most likely answer.

4 Pointer Networks

在该论文中将上面的 attention 的具体计算方法直接给出了，此为对第 i 个输出计算，e_i,d_i 分别便是 encoder 和 decoder 的 hidden states：

$$u_j^i = v^T \tanh(W_1 e_j + W_2 d_i), j \in (1, .., n)$$

$$a_j^i = softmax(u_j^i), j \in (1, ..., n)$$

$$d_i\text{'} = \sum_{j=1}^{n} a_j^i e_j$$

第二步的 softmax 即将 u_j^i 转化成 attention。

文中提出的 Ptr-Net 将 attention 的那一步直接替换成输入的每个 token 做为输出的概率，即：

$$u_j^i = v^T \tanh(W_1 e_j + W_2 d_i), j \in (1, .., n)$$

$$p(C_i | C_1, ..., C_{i-1}, P) = softmax(u^i)$$

这种结构在解决输出的内容在输入中出现非常好用。

5 Learning Natural Language Inference with LSTM

fig 此文本来是用来解决 natural language inference 的问题的，在上半部分是前人提出的 attention 机制，“A dog jumping for a Frisbee in the snow.” and the hypothesis “A cat washes his face and whiskers with his front paw.” 在该句子中 dog 和 cat 不匹配，应该被记住，最后得出两个句子关系为 contradiction，但是上面的结构直接由最终状态得出结论效果不好（距离太长容易忘记），所以受到 lstm 的启发，在向后传的时候增加了几个门，用于控制记忆和遗忘。正是用了这些门机制 (和 LSTM 的门非常类似，这里不再写出), 所以该结构称之为 Match-LSTM。

6 Machine Comprehension Using Match-LSTM and Answer Pointer

上图即此论文提出的结构，最下面是将问题过一个 LSTM，第二层是将原文过一次 LSTM，然后通过上文提到的 Match-LSTM 方法处理，再通过 Pointer Networks 获得输出，左半边是原来的 pointer networks，右边的是直接截取一个开始点和一个终止点得到答案。最终的 loss function 如下：

$$-\sum_{n=1}^{N} \log p(a_n | P_n, Q_n)$$

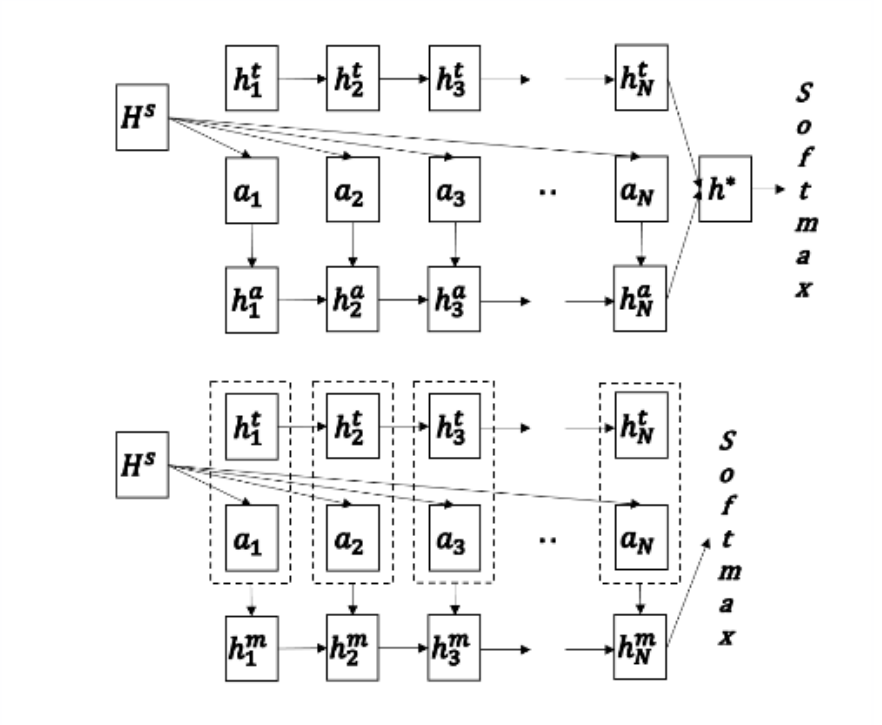

Figure 3: Match-LSTM

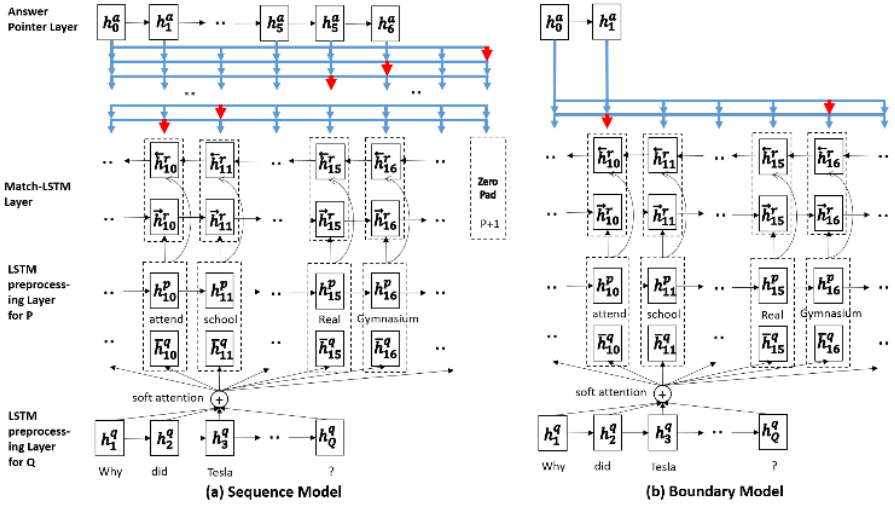

Figure 4: Match-LSTM

7 R-Net

有了之前的铺垫，R-net 就很好理解了，下面的 Question-Passage Matching(QPM) 就是之前提到的 Match-LSTM，中间加了一个 Passage Self-Matching，最后通过 Pointer Networks 预测输出。

R-net 的主要的创新点就在提出了 Passage Self-Matching，因为 QPM 只用到了问题和文章局部信息，并未挖掘文章本身信息之间的关联，所以加入了一个 self attention 之后效果很好。