论文阅读笔记

Step3

MF1833063, 史鹏, spwannasing@gmail.com

2019 年 6 月 24 日

1 MEMEN: Multi-layer Embedding with Memory Networks for Machine Comprehension

提出了新的模型：Multi-layer Embedding with Memory Network(MEMEN)。提出了多层embedding， encoding了word的句法以及语义信息。引入了full-orientation matching。

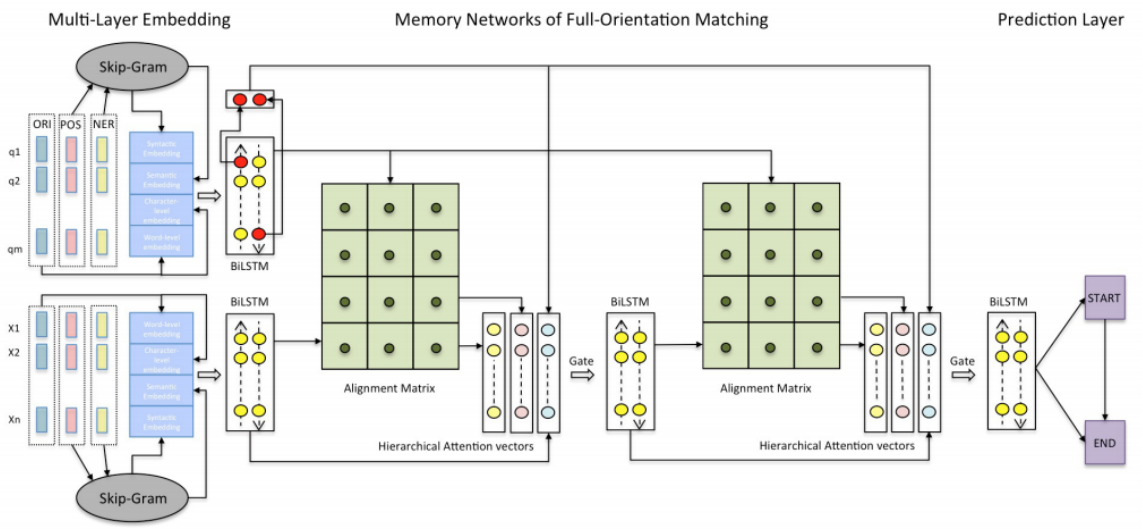


图 1: 网络结构图

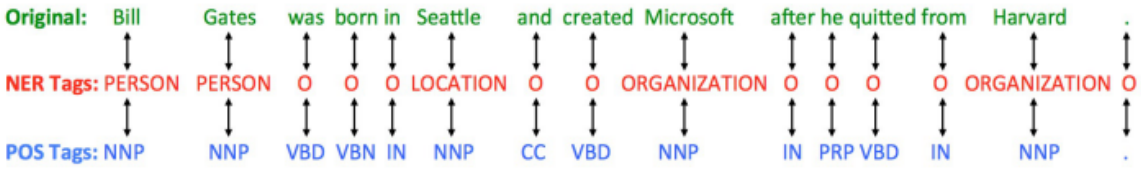


图 2: 三种信息的例子

1. Encoding of Context and Query

$$r_t^P = BiLSTM([w_t^P; c_t^P; s_t^P])$$
$$r_t^Q = BiLSTM([w_t^Q; c_t^Q; s_t^Q])$$

其中w，c，s分别代表word-level,character-level,tags的embedding，以及query的最后一个隐藏状态u^Q

2. Memory Network of Full-Orientation Matching

[1] Integral Query Matching

$$c_t = softmax(< u^Q, r_t^P >)$$

$$m^1 = \sum_t c_t r_t^P$$

[2] Query-Based Similarity Matching

首先得到alignment matrix $A \in \mathbb{R}^{n\times m}$,$A_{ij} = w_1^T[r_i^P; r_j^Q; r_i^P \circ r_j^Q]$

$$B = softmax_{row}(A) \in R^{n\times m}$$

M^2的每一列$M_t^2 = B \cdot r^Q$

[3] Context-Based Similarity Matching

$e = max_{row}(A) \int R^n$,the attention is $d = softmax(e)$

$$m^3 = \sum_t r_t^P \cdot d_t$$

$$M = f(M^1, M^2, M^3)$$

M^1, M^3是对应的向量重复n次。添加额外的gate

$$g_t = sigmoid(W_g M)$$

$$M^* = g_t \odot M$$

$$O_t = BiLSTM(O_{t-1}, M)$$

3. Output Layer

通过Pointer Network来预测答案。首先获取到该网络的初始hidden state：

$$z_j = s^T tanh(W^Q r_j^Q + b^Q)$$

$$a_i = \frac{exp(z_i)}{\sum exp(z_j)}$$

$$l_0 = \sum a_i r_i^Q$$

然后Pointer Network：

$$z_j^k = c^T tanh(W^P O_j + W^h l^0$$

$$a_i^k = \frac{exp(a_i^k)}{\sum exp(z_j^k)}$$

$$p^k = argmax(a_1^k, ..., a_n^k)$$

k=1,2

$$v^k = \sum_{i=1}^{n} a_i^k O_i$$

$$l_t^1 = GRU(l_{t-1}, v^k)$$

2 DYNAMIC COATTENTION NETWORKS FOR QUESTION ANSWERING

因为以前的一些方法都是single-pass，所以没有方法从局部最大恢出来。本篇文章提出了Dynamic Coattention Network (DCN)。首先是融合问题和文档的相互依赖的表示，以便将重点放在两者的相关部分上。然后Dynamic pointing decoder在潜在的答案范围上迭代。(我理解的这里所谓的迭代法其实就是通过LSTM来decoder)

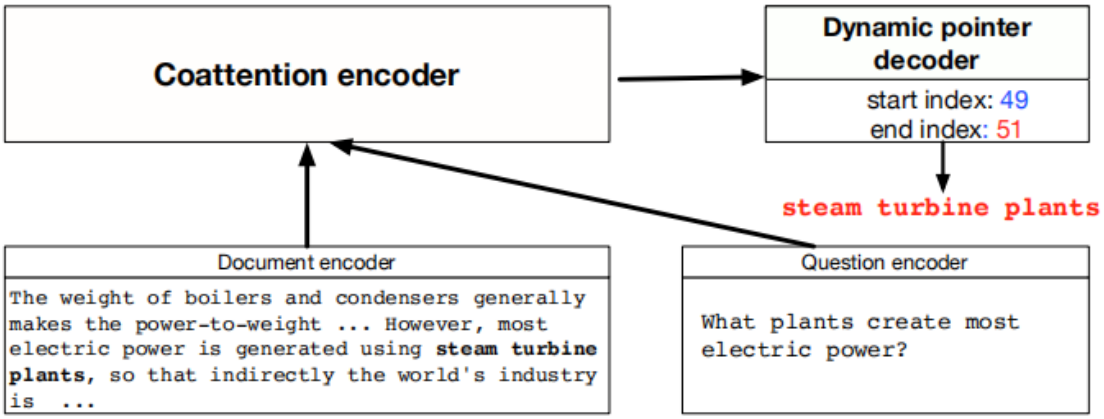


图 3: overview

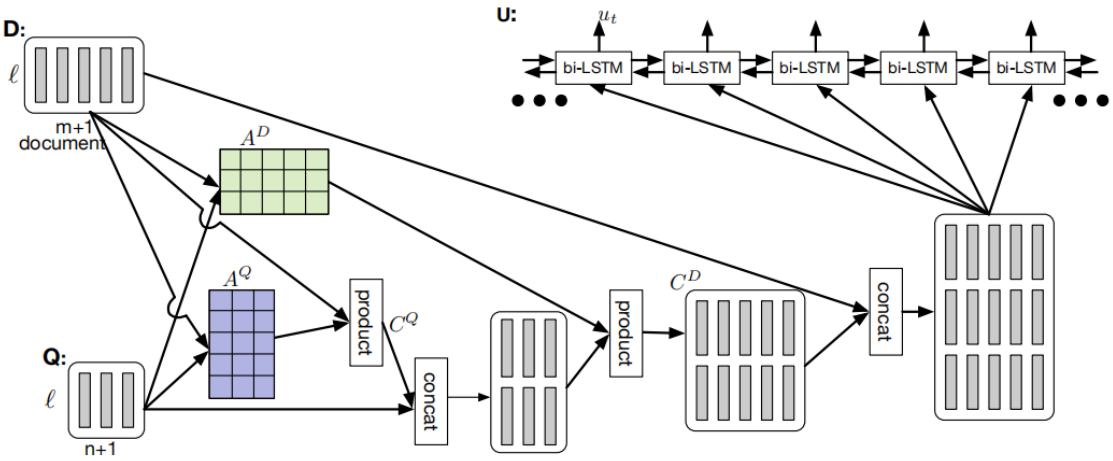


图 4: Coattention encoder.

1. CoAttention Encoder

首先计算相似度矩阵$L = D^T Q \in \mathbb{R}^{(m+1)\times(n+1)}$，然后得到对应的row-wise与column-wise：

$$A^Q = softmax(L) \in \mathbb{R}^{(m+1)\times(n+1)}$$

$$A^D = softmax(L^T) \in \mathbb{R}^{(n+1)\times(m+1)}$$

$$C^Q = DA^Q \in \mathbb{R}^{l\times(n+1)}$$

$$C^D = [Q; C^Q]A^D \in \mathbb{R}^{2l\times(m+1)}$$

这里定义C^D就是co-dependent representation of the question and document

$$u_t = Bi - LSTM(u_{t-1}, u_{t+1}, [d_d; c_t^D]) \in \mathbb{R}^{2l}$$

2. Dynamic Pointing Decoder

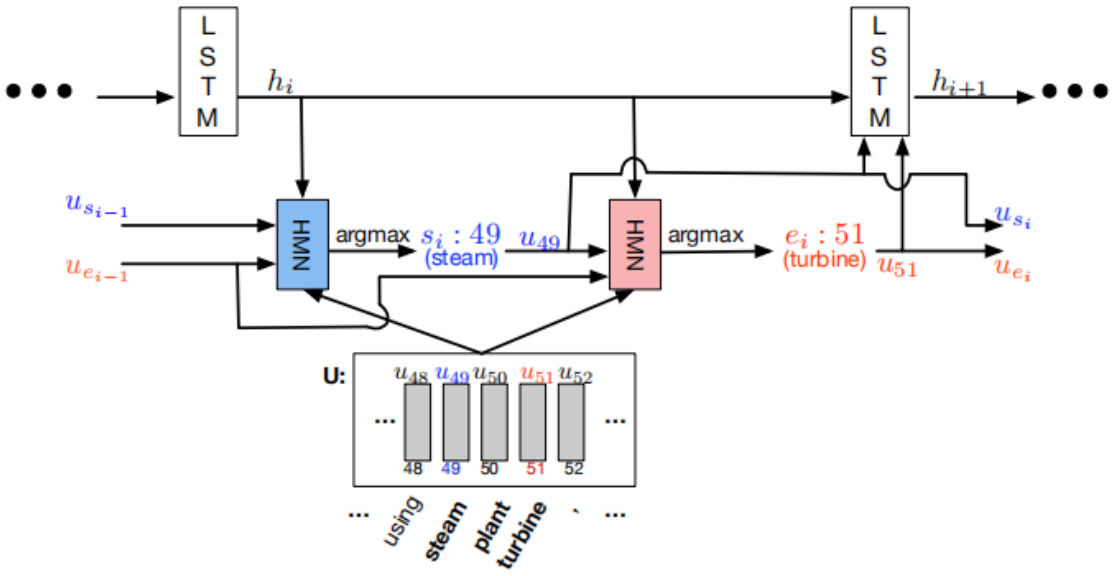


图 5: Dynamic Decode

在每一个step都会考虑到当前预测的start和end对应的co-attenion的encoding来更新自己的状态。

$$h_i = LSTM_{dec}(h_{i-1}, [u_{s_{i-1}}; u_{e_{i-1}}])$$

$$s_i = \underset{t}{argmax}(\alpha_1, ..., \alpha_m)$$

$$e_i = \underset{t}{argmax}(\beta_1, ..., \beta_m)$$

α, β的计算类似，下面只介绍α

$$\alpha_t = HMN_{start}(u_t, h_i, u_{s_{i-1}}, u_{e_{i-1}})$$

$$\begin{aligned}
\text{HMN}\left(u_t, h_i, u_{s_{i-1}}, u_{e_{i-1}}\right) &= \max\left(W^{(3)}\left[m_t^{(1)}; m_t^{(2)}\right] + b^{(3)}\right) \\
r &= \tanh\left(W^{(D)}\left[h_i; u_{s_{i-1}}; u_{e_{i-1}}\right]\right) \\
m_t^{(1)} &= \max\left(W^{(1)}\left[u_t; r\right] + b^{(1)}\right) \\
m_t^{(2)} &= \max\left(W^{(2)} m_t^{(1)} + b^{(2)}\right)
\end{aligned}$$

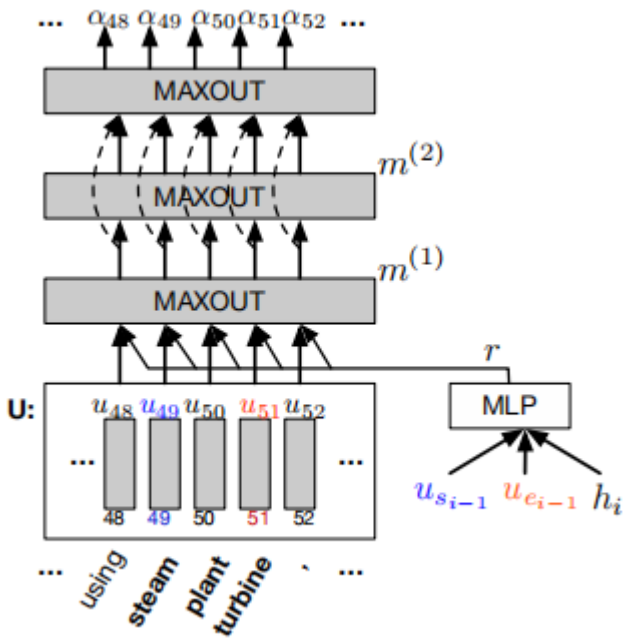
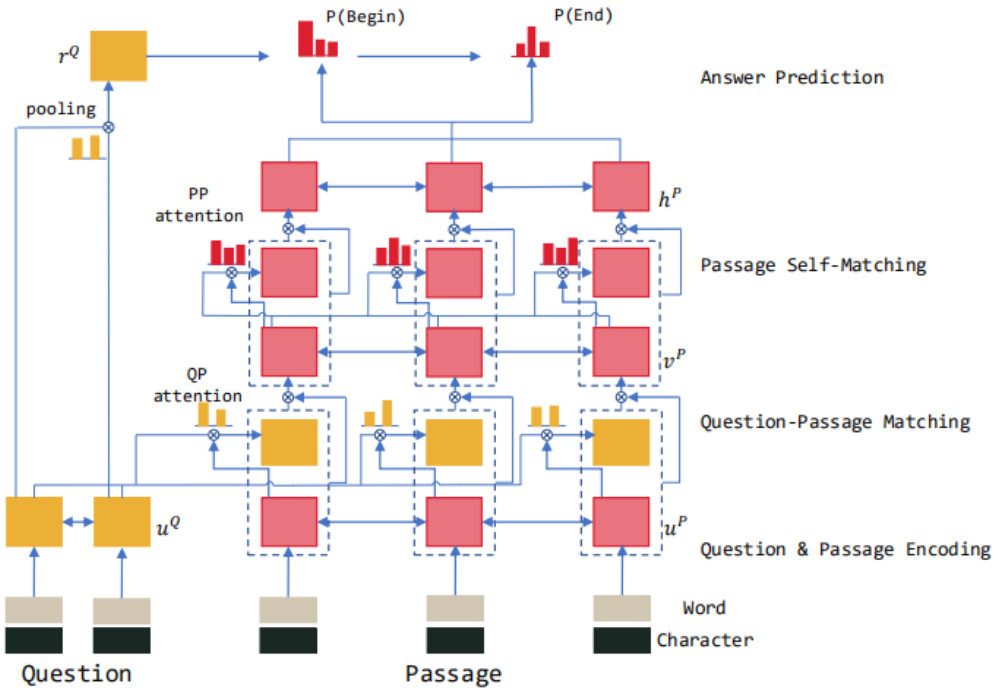


图 6: Highway Maxout Network

3 R-NET: MACHINE READING COMPREHENSION WITH SELF-MATCHING NETWORKS

首先得到question-aware的passage表示，然后提出一种self-matching 注意力机制抽取信息，最后通过pointer Network选择答案。



1. QUESTION AND PASSAGE ENCODER

$$u_t^Q = BiGRU_Q(u_{t-1}^Q, [e_t^Q, c_t^Q])$$
$$u_t^P = BiGRU_P(u_{t-1}^P, [e_t^P, c_t^P])$$

2. GATED ATTENTION-BASED RECURRENT NETWORKS

我们提出了一种基于门控注意的递归网络，将问题信息整合到段落表示中.它是一种基于注意力的递归网络的变体，有一个额外的门确定文章中有关问题的信息的重要性。

$$v_t^P = RNN(v_{t-1}^P, c_t)$$

here $c_t = att(u_Q, [u_t^P, v_{t-1}^P])$

$$s_j^t = v^T tanh(W_u^Q u_j^Q + W_u^P u_t^P + W_v^P v_{t-1}^P)$$
$$a_i^t = \frac{exp(s_i^t)}{\sum s_j^t}$$
$$c_t = \sum_{i=1}^{m} a_i^t u_i^Q$$

match-LSTM,将u_t^P作为额外的输入。

$$v_t^P = RNN(v_{t-1}^P, [u_t^p, c_t])$$

为了决定passage某部分的重要性以及其相关的question，加了一个额外的gate

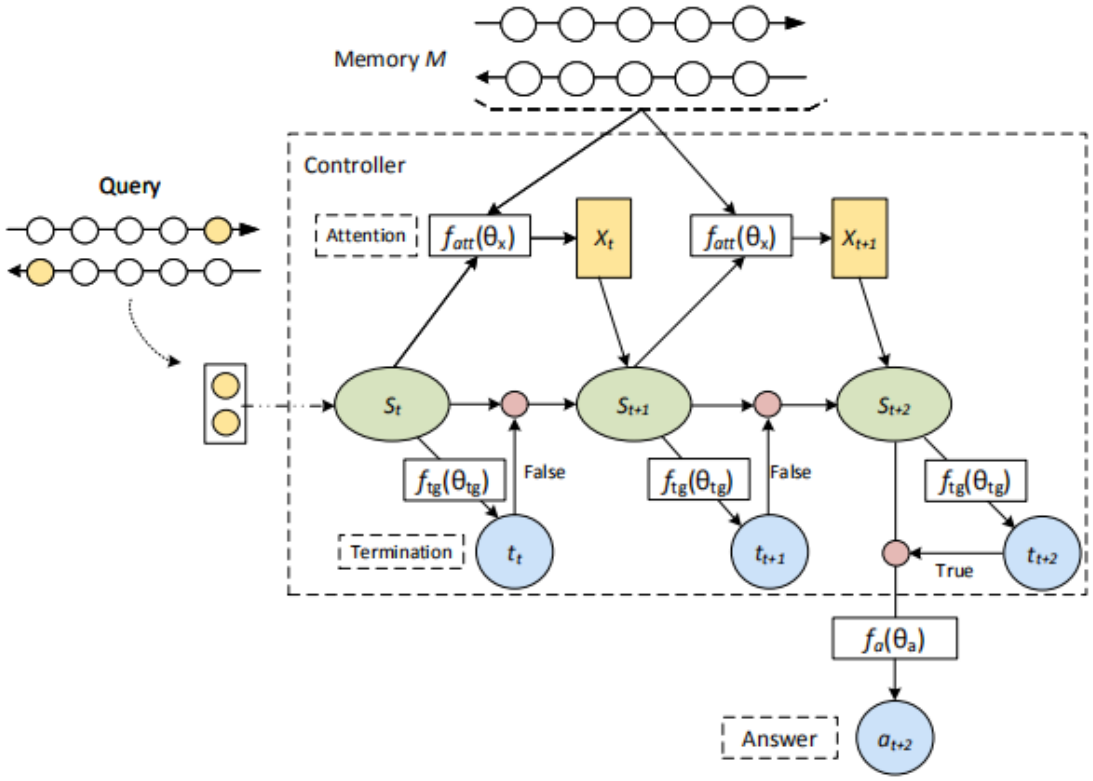
$$g_t = sigmoid(W_g[u_t^P, c_t])$$

$$[u_t^P, c_t]^* = g_t \odot [u_t^P, c_t]$$

3. 剩下的 SELF-MATCHING ATTENTION 以及OUTPUT LAYER都和之前的类似。

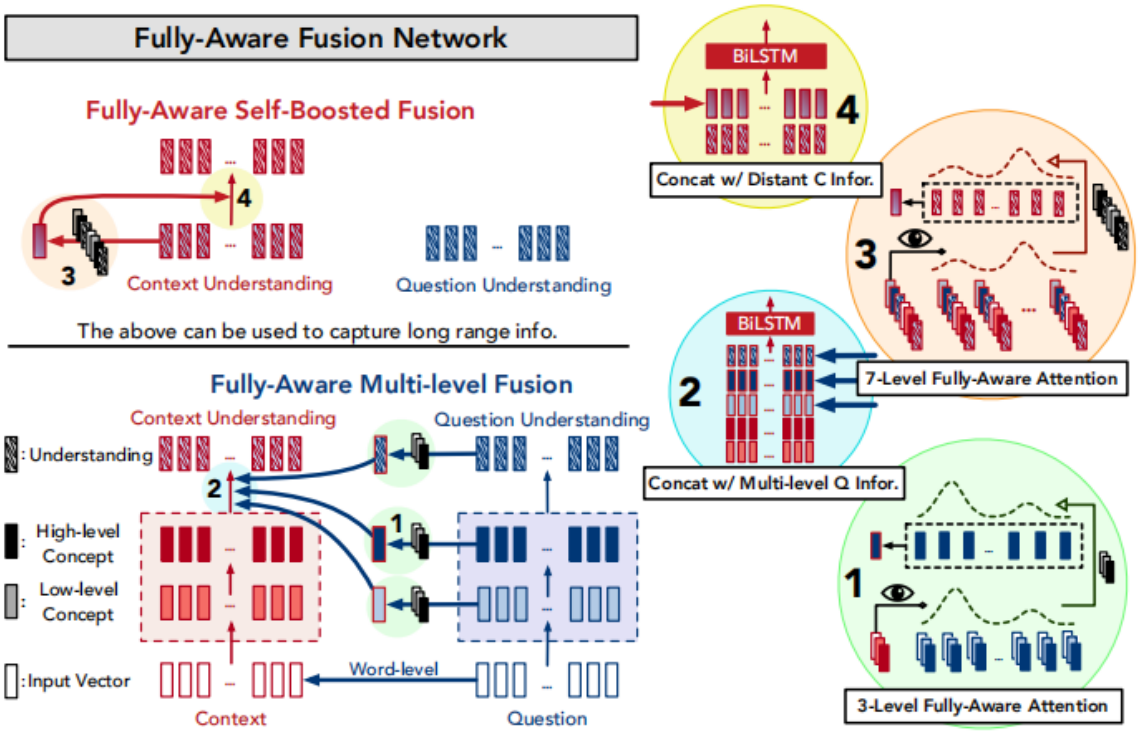
4 ReasoNet: Learning to Stop Reading in Machine Comprehension

引入强化学习，从而能够学得机器应该“阅读”多少次之后，停止迭代。只有正确答案的奖赏是1，其它都是0。



5 FUSIONNET: FUSING VIA FULLY-AWARE ATTENTION WITH APPLICATION TO MACHINE COMPREHENSION

提出了一个新的概念“history of word”,其实核心思想就是获取到更多的word的表示，将不同level的embedding和词性信息等以及经过神经网络计算后得到的向量都组合在一起，然后在Attention的环节，是计算一组矩阵之间的相似度而不是一组向量。



1. 首先定义history of the ith word 为HoW_i,它是把所有为这个单词生成的表示连接起来。

2. Fully-Aware Multi-level Fusion

   [1] Word-level

$$\hat{g}_i^C = \sum_j \alpha_{ij} g_j^Q$$

where $\alpha_{ij} \propto exp(S(g_i^C, g_j^Q)), S(x,y) = RELU(Wx)^T RELU(Wy)$

$$\tilde{w}_i^C = [w_i^C, em_i, \hat{g}_i^C]$$

$$\boldsymbol{h}_1^{Cl}, \ldots, \boldsymbol{h}_m^{Cl} = \text{BiLSTM}(\tilde{\boldsymbol{w}}_1^C, \ldots, \tilde{\boldsymbol{w}}_m^C), \quad \boldsymbol{h}_1^{Ql}, \ldots, \boldsymbol{h}_n^{Ql} = \text{BiLSTM}(\boldsymbol{w}_1^Q, \ldots, \boldsymbol{w}_n^Q),$$
$$\boldsymbol{h}_1^{Ch}, \ldots, \boldsymbol{h}_m^{Ch} = \text{BiLSTM}(\boldsymbol{h}_1^{Cl}, \ldots, \boldsymbol{h}_m^{Cl}), \quad \boldsymbol{h}_1^{Qh}, \ldots, \boldsymbol{h}_n^{Qh} = \text{BiLSTM}(\boldsymbol{h}_1^{Ql}, \ldots, \boldsymbol{h}_n^{Ql}).$$

$$U_Q = \{\boldsymbol{u}_1^Q, \ldots, \boldsymbol{u}_n^Q\} = \text{BiLSTM}([\boldsymbol{h}_1^{Ql}; \boldsymbol{h}_1^{Qh}], \ldots, [\boldsymbol{h}_n^{Ql}; \boldsymbol{h}_n^{Qh}]).$$

[2] Higher-level

$$\text{HoW}_i^C = [\boldsymbol{g}_i^C; \boldsymbol{c}_i^C; \boldsymbol{h}_i^{Cl}; \boldsymbol{h}_i^{Ch}], \ \text{HoW}_i^Q = [\boldsymbol{g}_i^Q; \boldsymbol{c}_i^Q; \boldsymbol{h}_i^{Ql}; \boldsymbol{h}_i^{Qh}] \in \mathbb{R}^{1400}$$

1. Low-level fusion: $\hat{\boldsymbol{h}}_i^{Cl} = \sum_j \alpha_{ij}^l \boldsymbol{h}_j^{Ql}, \quad \alpha_{ij}^l \propto \exp(S^l(\text{HoW}_i^C, \text{HoW}_j^Q))$.
2. High-level fusion: $\hat{\boldsymbol{h}}_i^{Ch} = \sum_j \alpha_{ij}^h \boldsymbol{h}_j^{Qh}, \quad \alpha_{ij}^h \propto \exp(S^h(\text{HoW}_i^C, \text{HoW}_j^Q))$.
3. Understanding fusion: $\hat{\boldsymbol{u}}_i^C = \sum_j \alpha_{ij}^u \boldsymbol{u}_j^Q, \quad \alpha_{ij}^u \propto \exp(S^u(\text{HoW}_i^C, \text{HoW}_j^Q))$.

$$\{\boldsymbol{v}_1^C, \ldots, \boldsymbol{v}_m^C\} = \text{BiLSTM}([\boldsymbol{h}_1^{Cl}; \boldsymbol{h}_1^{Ch}; \hat{\boldsymbol{h}}_1^{Cl}; \hat{\boldsymbol{h}}_1^{Ch}; \hat{\boldsymbol{u}}_1^C], \ldots, [\boldsymbol{h}_m^{Cl}; \boldsymbol{h}_m^{Ch}; \hat{\boldsymbol{h}}_m^{Cl}; \hat{\boldsymbol{h}}_m^{Ch}; \hat{\boldsymbol{u}}_m^C]).$$

3. Fully-Aware Self-Boosted Fusion.

$$\text{HoW}_i^C = [\boldsymbol{g}_i^C; \boldsymbol{c}_i^C; \boldsymbol{h}_i^{Cl}; \boldsymbol{h}_i^{Ch}; \hat{\boldsymbol{h}}_i^{Cl}; \hat{\boldsymbol{h}}_i^{Ch}; \hat{\boldsymbol{u}}_i^C; \boldsymbol{v}_i^C] \in \mathbb{R}^{2400}.$$

$$U_C = \{\boldsymbol{u}_1^C, \ldots, \boldsymbol{u}_m^C\} = \text{BiLSTM}([\boldsymbol{v}_1^C; \hat{\boldsymbol{v}}_1^C], \ldots, [\boldsymbol{v}_m^C; \hat{\boldsymbol{v}}_m^C])$$

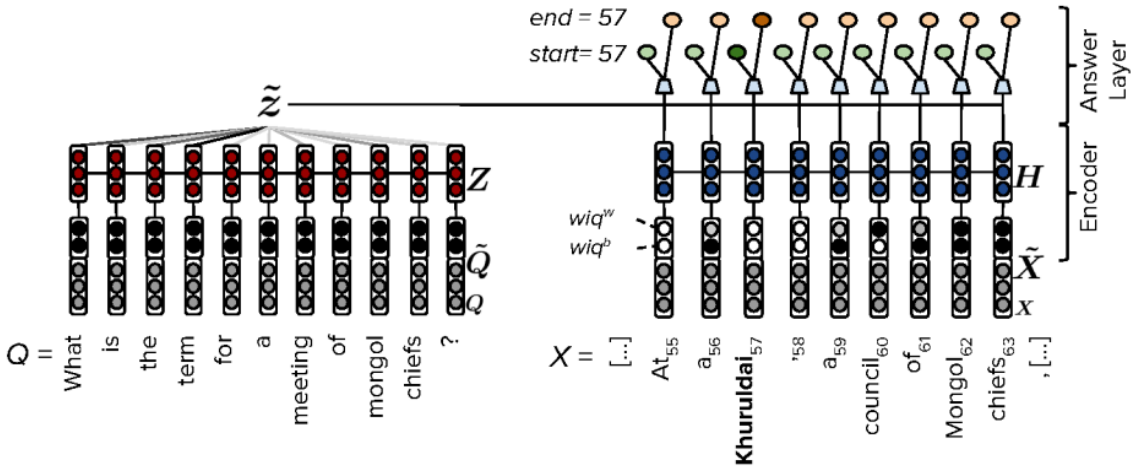
$$P_i^S \propto \exp((\boldsymbol{u}^Q)^T W_S \boldsymbol{u}_i^C),$$

$$P_i^E \propto \exp((\boldsymbol{v}^Q)^T W_E \boldsymbol{u}_i^C)$$

6 Making Neural QA as Simple as Possible but not Simpler

典型的神经结构由embedding、encoding、interaction和answer层组成。在本工作中，我们使用上下文/类型匹配启发式作为准则，为抽取QA任务导出简单的神经基线体系结构,提出了基于BoW和BiRNN的baseline，叫做FastQA。

【注】启发式算法可以这样定义：一个基于直观或经验构造的算法，在可接受的花费（指计算时间和空间）下给出待解决组合优化问题每一个实例的一个可行解，该可行解与最优解的偏离程度一般不能被预计。



【注】wiq 是指word-in-question feeatures

7 Efficient and Robust Question Answering from Minimal Context over Documents

研究回答问题所需的最少的context，发现大多数数据集中的问题能够用少量的句子做出回答。因此，提出了一个句子选择器来选出送入QA模型的最小的句子集合。我们的句子选择器利用了三种简单的技术-权重传递、数据修改和分数归一化，这些技术在句子选择的任务上是非常有效的。

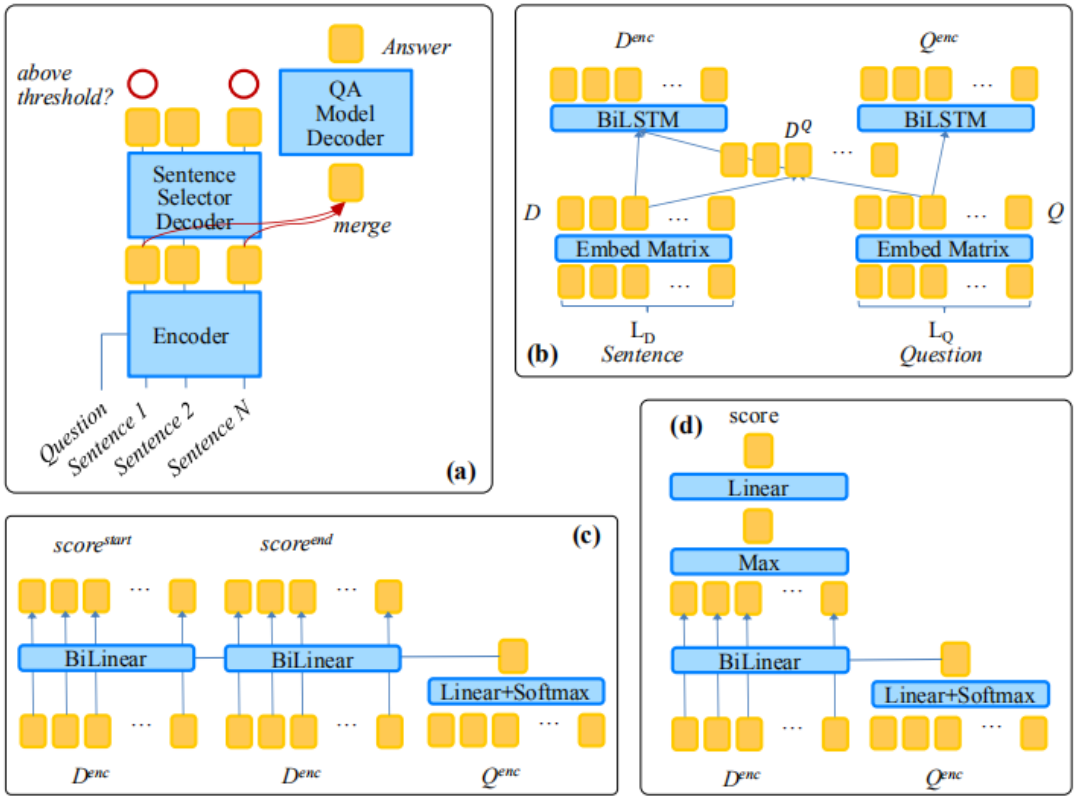


Figure 2: Our model architecture. (a) Overall pipeline, consisting of sentence selector and QA model. Selection score of each sentence is obtained in parallel, then sentences with selection score above the threshold are merged and fed into QA model. (b) Shared encoder of sentence selector and S-Reader (QA Model), which takes document and the question as inputs and outputs the document encodings D^{enc} and question encodings Q^{enc}. (c) Decoder of S-Reader (QA Model), which takes D^{enc} and Q^{enc} as inputs and outputs the scores for start and end positions. (d) Decoder of sentence selector, which takes D^{enc} and Q^{enc} for each sentence and outputs the score indicating if the question is answerable given the sentence.

图 7: 整体结构图

句子选择器并行计算每个句子的选择分数。我们向QA模型提供了一组选择分数较高的简化句子来回答问题，分数代表是否能根据这句话来回答问题。模型分为encoder模块和decoder模块。

(a) encoder:计算句子和问题的编码。

[1] 分别计算sentence embedding $D \in \mathbb{R}^{h_d \times L_d}$， question embedding $Q \in \mathbb{R}^{h_d \times L_q}$，以及question-aware sentence embedding $D^q \in \mathbb{R}^{h_d \times L_d}$。其中question-aware sentence embedding通过以下方式计算：

$$\alpha_i = \text{softmax}\left(D_i^T W_1 Q\right) \in \mathbb{R}^{L_q}$$
$$D_i^q = \sum_{j=1}^{L_q} \left(\alpha_{i,j} Q_j\right) \in \mathbb{R}^{h_d}$$

[2]

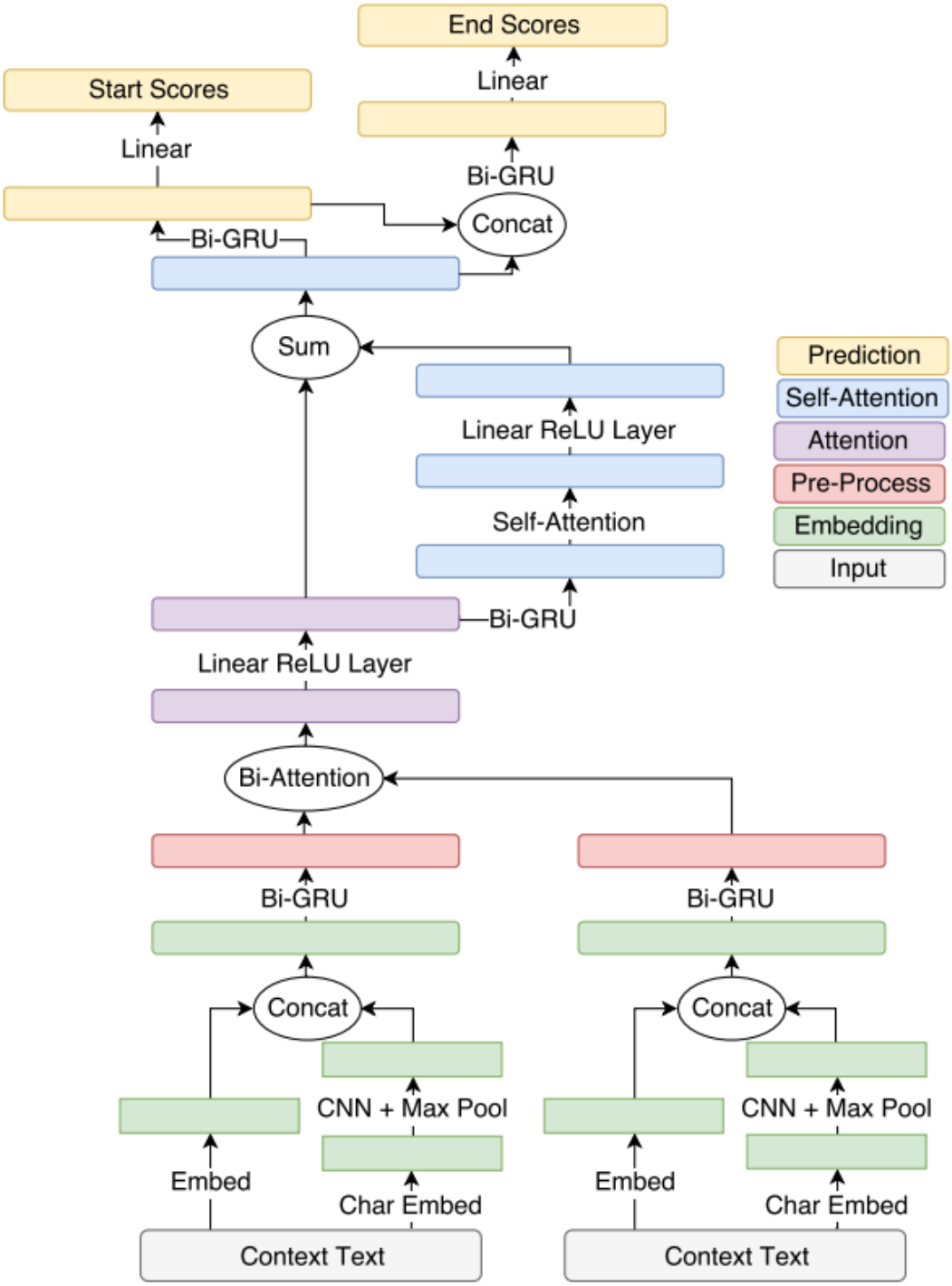
$$D^{enc} = \text{BiLSTM}\left([D_i; D_i^q]\right) \in \mathbb{R}^{h \times L_d}$$
$$Q^{enc} = \text{BiLSTM}\left(Q_j\right) \in \mathbb{R}^{h \times L_q}$$

(b) dercoder是一个特定于任务的模块，它通过计算句子编码和问题编码之间的双线性相似性来计算句子的得分，如下所示:

[1]

$$\beta = \text{softmax}\left(w^T Q^{enc}\right) \in \mathbb{R}^{L_q}$$
$$q^{\tilde{e}nc} = \sum_{j=1}^{L_q} \left(\beta_j Q_j^{enc}\right) \in \mathbb{R}^h$$
$$\tilde{h}_i = \left(D_i^{enc} W_2 q^{\tilde{e}nc}\right) \in \mathbb{R}^h$$
$$\tilde{h} = \max\left(\tilde{h_1}, \tilde{h_2}, \cdots, h_{L_d}\right)$$
$$\text{score} = W_3^T \tilde{h} \in \mathbb{R}^2$$

8 Simple and Effective Multi-Paragraph Reading Comprehension



我们介绍了一种将段落级问答模型应用于整个文档作为输入的情况的方法。我们表明，使用一种改进的训练方案可以显着地提高模型的性能，该方案使模型忽略包含非答案的段落。我们的方法包括从每个文档中抽取多个段落，并使用一个要求模型生成全局正确输出的目标函数。

(a) Pipelined Method-选择一个段落并将其传递给段落级的问答模型。

(b) Condfidence Method-我们使用非归一化和非指数(即在应用Softmax运算符之前)对每个跨度的评分作为模型的度量，将该模型适用于多段设置。

9 NEURAL SPEED READING VIA SKIM-RNN

提出了一种新的递归神经网络(RNN)，它动态地决定只对相对不重要的输入令牌更新隐藏状态的一小部分。

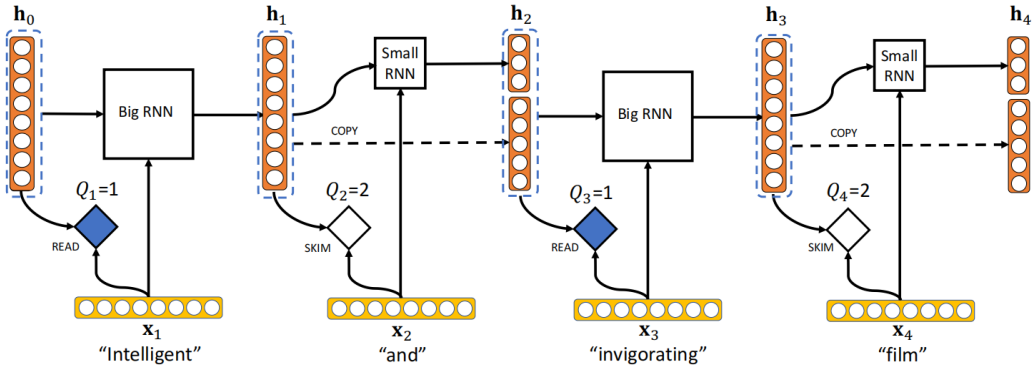


Figure 1: The schematic of Skim-RNN on a sample sentence from Stanford Sentiment Treebank: "intelligent and invigorating film". At time step 1, Skim-RNN makes the decision to *read* or *skim* $\mathbf{x}_1$ by using Equation 1 on $\mathbf{h}_0$ and $\mathbf{x}_1$. Since 'intelligent' is an important word for sentiment, it decides to *read* (blue diamond) by obtaining a full-size hidden state with the big RNN and updating the entire previous hidden state. At time step 2, Skim-RNN decides to *skim* (empty diamond) the word 'and' by updating the first few dimensions of the hidden state using small RNN.

$$\mathbf{p}_t = \text{softmax}\left(\alpha\left(\mathbf{x}_t, \mathbf{h}_{t-1}\right)\right) = \text{softmax}\left(\mathbf{W}\left[\mathbf{x}_t; \mathbf{h}_{t-1}\right] + \mathbf{b}\right) \in \mathbb{R}^k$$

$$Q_t \sim \text{Multinomial}\left(\mathbf{p}_t\right)$$

$$\mathbf{h}_t = \begin{cases} f\left(\mathbf{x}_t, \mathbf{h}_{t-1}\right), & \text{if } Q_t = 1 \\ \left[f'\left(\mathbf{x}_t, \mathbf{h}_{t-1}\right); \mathbf{h}_{t-1}\left(d'+1:d\right)\right], & \text{if } Q_t = 2 \end{cases}$$

$$\mathbb{E}_{Q_t \sim \text{Multinomial}(\mathbf{p}_t)}[L(\theta)] = \sum_Q L(\theta; Q) P(Q) = \sum_Q L(\theta; Q) \prod_j \mathbf{p}_j^{Q_j}$$

$$\mathbf{r}_t^i = \frac{\exp\left(\left(\log\left(\mathbf{p}_t^i\right) + g_t^i\right)/\tau\right)}{\sum_j \exp\left(\left(\log\left(\mathbf{p}_t^j\right) + g_t^j\right)/\tau\right)}$$

$$\mathbf{h}_t = \sum_i \mathbf{r}_t^i \tilde{\mathbf{h}}_t^i$$

$$\tilde{\mathbf{h}}_t^1 = f\left(\mathbf{x}_t, \mathbf{h}_{t-1}\right)$$

$$\tilde{\mathbf{h}}_t^2 = \left[f'\left(\mathbf{x}_t, \mathbf{h}_{t-1}\right); \mathbf{h}_{t-1}\left(d'+1:d\right)\right]$$

10 Hierarchical Attention Flow for Multiple-Choice Reading Comprehension

提出了一个hierarchical attention flow结构，利用候选option的信息来进一步挖掘passage的隐藏信息。

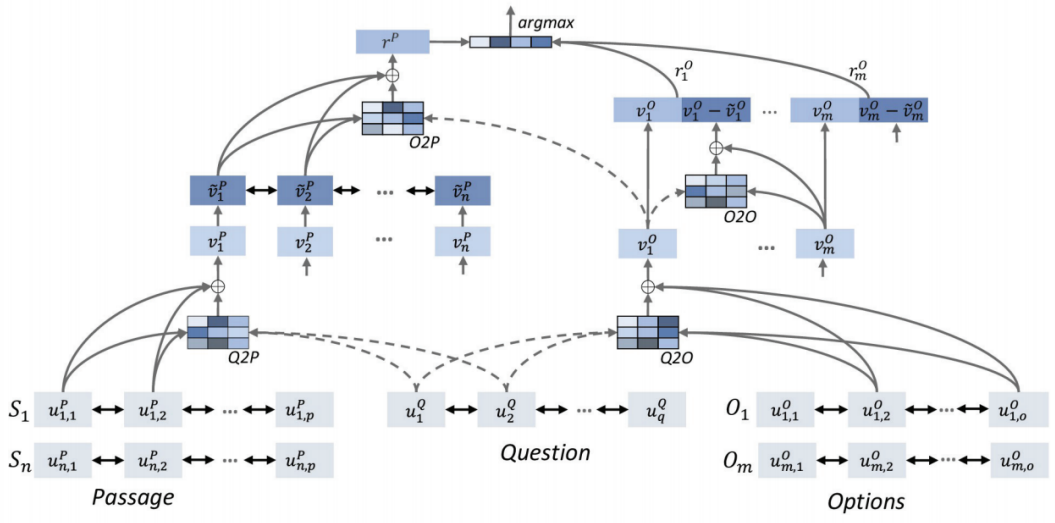


Figure 2: Hierarchical Attention Flow overview. The superscripts P, Q and O correspond to passage, question and option respectively. The text blocks marked with the same color are at the same hierarchical level. $\oplus$ represents the weighted sum operation. Finally, *argmax* function outputs the option with highest score as the answer.

1 Word Context Encoder

$$\begin{aligned} u_t^Q &= \mathrm{BiGRU}_Q\left(u_{t-1}^Q, e_t^Q\right) \\ u_{i,t}^P &= \mathrm{BiGRU}_P\left(u_{i,t-1}^P, e_{i,t}^P\right) \\ u_{i,t}^O &= \mathrm{BiGRU}_O\left(u_{i,t-1}^O, e_{i,t}^O\right) \end{aligned} \tag{10.1}$$

2 Attention Mechanism

$$\begin{aligned} A_{i,j} &= X_i W_{xy} Y_j \\ s_{i,j} &= \frac{\exp(A_{i,j})}{\sum_j^n \exp(A_{i,j})} \\ a_j &= \frac{1}{m}\sum_{i=1}^m s_{i,j} \end{aligned} \tag{10.2}$$

3 Question-to-Passage (Q2P) Word-level Attention

$$\begin{aligned} a &= att\left(u^Q, u_i^P | W_{qp}\right) \\ v_i^P &= \sum_t a_t u_{i,t}^P \end{aligned} \tag{10.3}$$

4 Question-to-Option (Q2O) Word-level Attention

$$\begin{aligned} a &= \mathrm{att}\left(u^Q, u_i^O | W_{qo}\right) \\ v_i^O &= \sum_t a_t u_{i,t}^O \end{aligned} \tag{10.4}$$

5 Sentence Context Encoder

$$\tilde{v}_i^P = \mathrm{BiGRU}_S\left(\tilde{v}_{i-1}^P, v_i^P\right) \tag{10.5}$$

6 Option-to-Passage (O2P) Sentence-level Attention

$$\begin{aligned} a &= att\left(v^O, \tilde{v}^P | W_{op}\right) \\ r^P &= \sum a_i \tilde{v}_i^P \end{aligned} \tag{10.6}$$

7 Option Correlations

$$\begin{aligned} A_{i,j} &= v_i^O W_{oo} v_j^O \\ s_{i,j} &= \frac{1(i \neq j)\exp(A_{i,j})}{\sum_j 1(i \neq j)\exp(A_{i,j})} \\ \tilde{v}_i^O &= \sum_j s_{i,j} v_j^O \\ r_i^O &= \left[v_i^O; v_i^O - \tilde{v}_i^O\right] \end{aligned} \tag{10.7}$$

8 answer prediction

$$\begin{aligned} s_i &= r^P W_p r_i^O \\ p_i &= \frac{\exp(s_i)}{\sum_j \exp(s_j)} \\ \text{ans} &= \text{argmax}_i\left(p_i\right) \end{aligned} \tag{10.8}$$

11 Towards Reading Comprehension for Long Document

利用hierarchical—LSTM来学习到paragraph-level的文本表示，从而能够从document中提取答案span，利用匹配机制找出最有可能含有答案的paragraph。核心就是将每一个paragraph生成一个向量，从而形成paragraph级的文本表示，然后再进行选择。

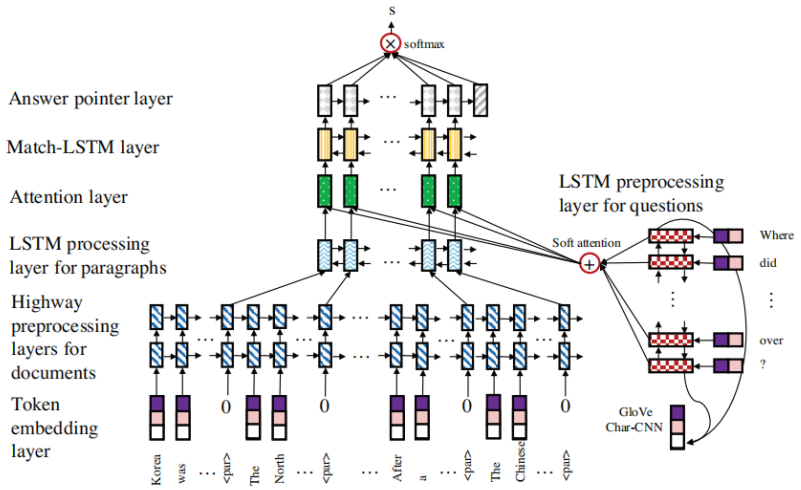


Figure 2: The structure of the MH-LSTM model. The model takes the word-level embeddings and the character-level embeddings as input. A two-layer highway network is used to learn the representations of each word in documents, and a bi-LSTM model to learn the representations of queries. The representations of each paragraph are extracted by mask, which are then fed into a match-LSTM network. The outputs are generated by a softmax pointer.

1. Token Embedding Layer

2. Preprocessing Layer for Query

$$\mathbf{H}^Q, \mathbf{q}_{rep} = \overleftrightarrow{\text{LSTM}}\,(Q) \tag{11.1}$$

3. Preprocessing Layer for Context

我们将q_{rep}连接到文档中每个单词的嵌入构成p_t。

$$\begin{aligned}
\mathbf{s}_{l,t}^P &= \mathbf{h}_{l,t}^P \cdot \mathbf{t}_{l,t}^P + \mathbf{s}_{l-1,t}^P \cdot \mathbf{c}_{l,t}^P \\
\mathbf{h}_{l,t}^P &= \tanh\left(\mathbf{W}_H^P \mathbf{p}_t \mathbb{I}_{\{l=1\}} + \mathbf{R}_{H_l}^P \mathbf{s}_{l-1,t}^P + \mathbf{b}_{H_l}^P\right) \\
\mathbf{t}_{l,t}^P &= \sigma\left(\mathbf{W}_T^P \mathbf{p}_t \mathbb{I}_{\{l=1\}} + \mathbf{R}_{T_l}^P \mathbf{s}_{l-1,t}^P + \mathbf{b}_{T_l}^P\right) \\
\mathbf{c}_{l,t}^P &= \sigma\left(\mathbf{W}_C^P \mathbf{p}_t \mathbb{I}_{\{l=1\}} + \mathbf{R}_{C_l}^P \mathbf{s}_{l-1,t}^P + \mathbf{b}_{C_l}^P\right) \\
\mathbf{H}^P &= \left[\mathbf{s}_{2,1}^P, \ldots, \mathbf{s}_{2,|P|}^P\right] \in \mathbb{R}^{h^P \times |P|}
\end{aligned} \tag{11.2}$$

4. Hierarchical Mask Layer

$$\mathbf{H}^M = \overleftrightarrow{\text{LSTM}}\left(\text{mask}\left(\mathbf{H}^P\right)\right) = \left[\mathbf{y}_1^M, \ldots, \mathbf{y}_{|M|}^M\right] \tag{11.3}$$

5. Attention Layer

$$\begin{aligned}
\vec{\mathbf{G}}_{t,i} &= \tanh\left(\overrightarrow{\mathbf{W}^{AQ}}\mathbf{y}_i^Q + \overrightarrow{\mathbf{W}^{AM}}\mathbf{y}_t^M + \overrightarrow{\mathbf{b}^A}\right) \\
\vec{\alpha}_t &= \text{softmax}\left(\vec{\mathbf{w}}^\top \vec{\mathbf{G}}_{t,*}\right)
\end{aligned} \tag{11.4}$$

6. Match-LSTM Layer

$$\begin{aligned}\overrightarrow{\mathrm{y}}_t &= \overrightarrow{\mathrm{LSTM}}\left(\overrightarrow{\mathrm{z}}_t, \overrightarrow{\mathrm{y}}_{t-1}\right)\\ \overrightarrow{\mathrm{z}}_t &= \begin{bmatrix}\mathbf{y}_t^M\\ \mathbf{H}^Q\vec{\alpha}_t^\top\end{bmatrix}\\ \overrightarrow{\mathbf{G}}_{t,i} &= \tanh\left(\overrightarrow{\mathbf{W}^{AQ}}\mathbf{y}_i^Q + \overrightarrow{\mathbf{W}^{AM}}\mathbf{y}_t^M + \overrightarrow{\mathbf{W}^{AO}}\mathbf{y}_t + \overrightarrow{\mathbf{b}^A}\right)\end{aligned} \tag{11.5}$$

7. Output Pointer Layer

$$\begin{aligned}\left[\mathbf{y}_1^e, \ldots, \mathbf{y}_{|M|}^e\right], \mathbf{e}_{rep} &= \text{ ENCODER }\left(\mathbf{H}^O\right)\\ \mathbf{y}^d &= \mathrm{DECODER}\left(\mathbf{e}_{rep}\right)\end{aligned} \tag{11.6}$$

8. Training

$$L(\theta) = -\frac{1}{N}\sum_{n=1}^{N}\log p\left(\widetilde{a_n}|\widetilde{P_n}, \widetilde{Q_n}\right) = -\frac{1}{N}\sum_{n=1}^{N}\log \mathbf{s}_{\widetilde{a_n}|\widetilde{P_n},\widetilde{Q_n}} \tag{11.7}$$

12 Joint Training of Candidate Extraction and Answer Selection for Reading Comprehension

大多数方法以独立的方式对答案进行建模，而忽略了它与其他candidate之间的关系。本文提出了一个两阶段的阅读理解框架：extract-then-select。我们首先从段落中提取答案候选人，然后结合所有candidata的信息选择最终答案，此外，我们将候选提取作为一个潜在变量，将两阶段过程与强化学习结合起来进行训练。

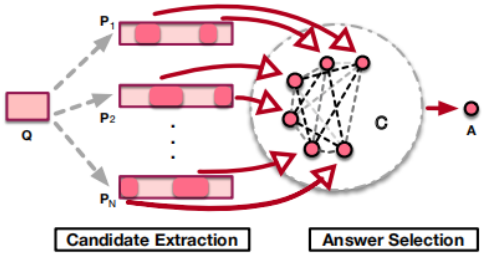


Figure 1: Two-stage RC Framework. The first part extracts candidates (denoted with circles) from all the passages. The second part establishes interactions among all these candidates to select the final answer. The different gray scales of dashed lines between candidates represent different intensities of interactions.

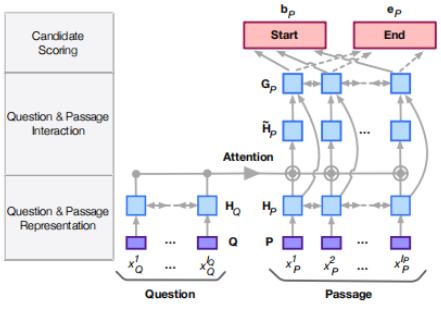


Figure 2: Candidate Extraction Model Architecture.

(a)

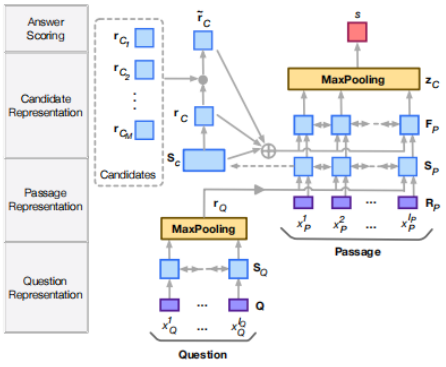


Figure 3: Answer Selection Model Architecture.

(b)

13 Multi-Passage Machine Reading Comprehension with Cross-Passage Answer Verification

multi-passage相比single Passage更加具有挑战性。我们提出了一种端到端的神经模型，使来自不同段落的答案候选人能够根据内容表示相互验证。我们联合训练了三个模块，它们可以根据三个因素来预测最终答案：答案边界、答案内容和交叉答案验证。

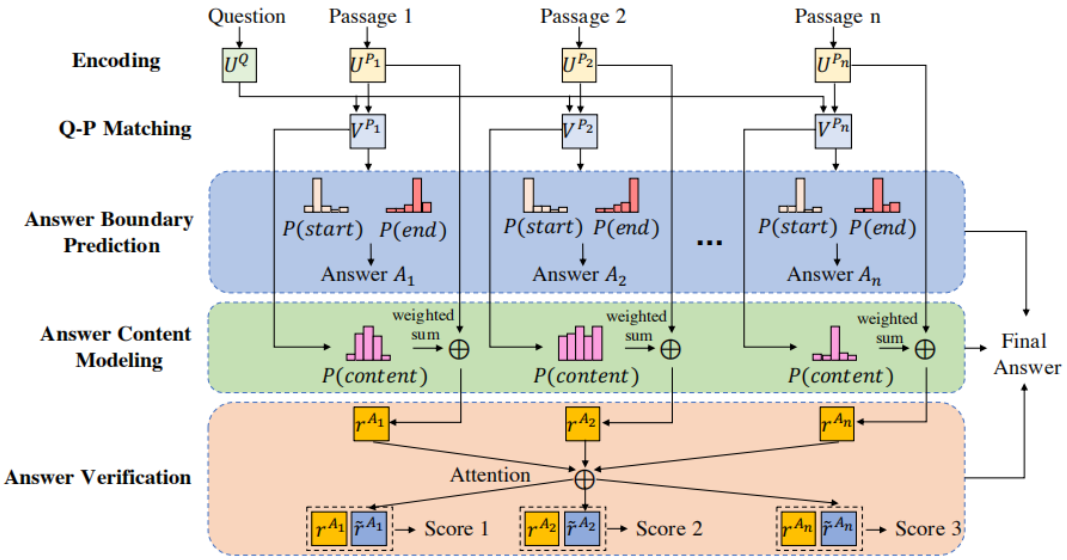


Figure 1: Overview of our method for multi-passage machine reading comprehension

1. Answer Boundary Prediction

$$
\begin{aligned}
g_k^t &= \mathbf{w}_1^{a\mathrm{T}} \tanh\left(\mathbf{W}_2^a \left[\mathbf{v}_k^P, \mathbf{h}_{t-1}^a\right]\right) \\
\alpha_k^t &= \exp\left(g_k^t\right) / \sum_{j=1}^{|\mathbf{P}|} \exp\left(g_j^t\right) \\
\mathbf{c}_t &= \sum_{k=1}^{|\mathbf{P}|} \alpha_k^t \mathbf{v}_k^P \\
\mathbf{h}_t^a &= \mathrm{LSTM}\left(\mathbf{h}_{t-1}^a, \mathbf{c}_t\right)
\end{aligned}
\tag{13.1}
$$

2. Answer Content Modeling

$$
p_k^c = \mathrm{sigmoid}\left(\mathbf{w}_1^{c\top} \mathrm{ReLU}\left(\mathbf{W}_2^c \mathbf{v}_k^{P_i}\right)\right) \tag{13.2}
$$

$$
\mathbf{r}^{A_i} = \frac{1}{|\mathbf{P}_i|} \sum_{k=1}^{|\mathbf{P}_i|} p_k^c \left[\mathbf{e}_k^{P_i}, \mathbf{c}_k^{P_i}\right] \tag{13.3}
$$

3. Cross-Passage Answer Verification

$$
s_{i,j} = \begin{cases} 0, & \text{if } i = j \\ \mathbf{r}^{A_i\top} \cdot \mathbf{r}^{A_j}, & \text{otherwise} \end{cases} \tag{13.4}
$$

$$\begin{aligned}\alpha_{i,j} &= \exp\left(s_{i,j}\right) / \sum_{k=1}^{n} \exp\left(s_{i,k}\right) \\ \tilde{\mathbf{r}}^{A_i} &= \sum_{j=1}^{n} \alpha_{i,j} \mathbf{r}^{A_j}\end{aligned} \tag{13.5}$$

$$g_i^v = \mathbf{w}^{v\top}\left[\mathbf{r}^{A_i}, \tilde{\mathbf{r}}^{A_i}, \mathbf{r}^{A_i} \odot \tilde{\mathbf{r}}^{A_i}\right] \tag{13.6}$$

$$p_i^v = \exp\left(g_i^v\right) / \sum_{j=1}^{n} \exp\left(g_j^v\right) \tag{13.7}$$

14 Reinforced Mnemonic Reader for Machine Reading Comprehension

首先提出了一种reattention机制，能够访问临时存储的multi-round对齐的过去的attentions，来细化当前的注意力。另外提出了一种新的优化方式，叫做dynamic-critical reinforcement learning，它总是鼓励预测一个更可接受的答案，以解决传统的强化学习算法中存在的收敛抑制问题。

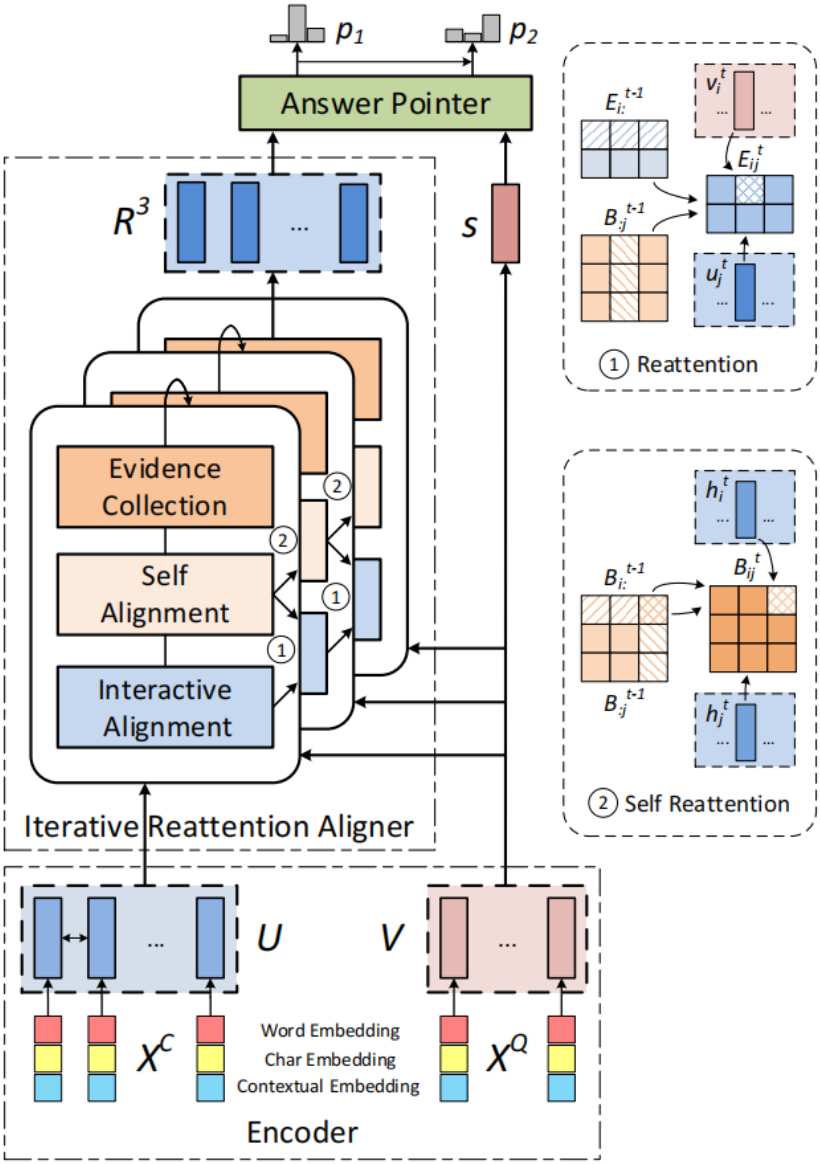


Figure 3: The architecture overview of Reinforced Mnemonic Reader. The subfigures to the right show detailed demonstrations of the reattention mechanism: 1) refined E^t to attend the query; 2) refined B^t to attend the context.

1. Reattention Mechanism

$$\begin{aligned}\tilde{E}_{ij}^{t} &= \text{softmax}\left(E_{i:}^{t-1}\right)\cdot\text{softmax}\left(B_{:j}^{t-1}\right)\\ E_{ij}^{t} &= f\left(v_i^t, u_j^t\right)+\gamma\tilde{E}_{ij}^{t}\end{aligned} \tag{14.1}$$

$$\begin{aligned}\tilde{B}_{ij}^{t} &= \text{softmax}\left(B_{i:}^{t-1}\right)\cdot\text{softmax}\left(B_{:j}^{t-1}\right)\\ B_{ij}^{t} &= \mathbb{1}_{(i\neq j)}\left(f\left(h_i^t, h_j^t\right)+\gamma\tilde{B}_{ij}^{t}\right)\end{aligned} \tag{14.2}$$

2. Dynamic-critical Reinforcement Learning

3. End-to-end Architecture

$$\begin{aligned}\tilde{x} &= \text{relu}\left(W_r[x;y;x\circ y;x-y]\right)\\ g &= \sigma\left(W_g[x;y;x\circ y;x-y]\right)\\ o &= g\circ\tilde{x}+(1-g)\circ x\end{aligned} \tag{14.3}$$

$$\begin{aligned}R^1, Z^1, E^1, B^1 &= \text{align}^1(U,V)\\ R^2, Z^2, E^2, B^2 &= \text{align}^2\left(R^1, V, E^1, B^1\right)\\ R^3, Z^3, E^3, B^3 &= \text{align}^3\left(R^2, V, E^2, B^2, Z^1, Z^2\right)\end{aligned} \tag{14.4}$$

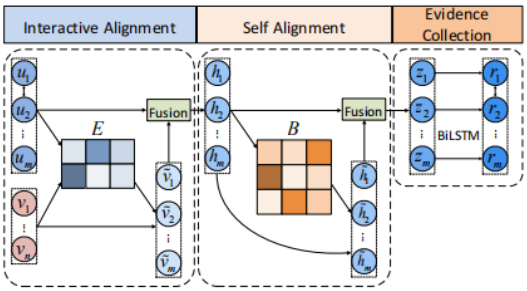


Figure 4: The detailed overview of a single aligning block. Different colors in E and B represent different degrees of similarity.