Question-Answering

Stanford Attentive Reader:

Encoder:

Bi-LSTM

Apply encoding to context

$\mathbf{p} = \{p_1, p_2, \ldots, p_m\}$ through a bi-LSTM network.

$\tilde{h}$ → hidden size.

$$\overrightarrow{h_i} = LSTM(\overrightarrow{h_{i-1}}, p_i), \quad i = 1, \ldots, m$$

$$\overleftarrow{h_i} = LSTM(\overleftarrow{h_{i-1}}, p_i), \quad i = m, \ldots, 1$$

and $\tilde{p_i} = concat(\overrightarrow{h_i}, \overleftarrow{h_i}) \in \mathbb{R}^h$

where $h = 2\tilde{h}$

The resulting forward and o/p vectors $\overrightarrow{H_p} = \{(h_p)_i\}$ and

$$\overleftarrow{H_p} = \{\overleftarrow{(h_p)_i}\}$$ are

concatenated into a final paragraph representation

$$(H_p)_i = \left(\overrightarrow{(H_p)_i}, \overleftarrow{(H_p)_i}\right).$$

Each o/p vector is chosen to be in $R^{\tilde{h}}$ $\left(\underline{\tilde{h} = 200}\right)$. Thus,

concatenation is in $\underline{R^{2\tilde{h}}}$ $\left(R^{h}\right)$

SUNDAY

Another (bi-LSTM or bi-GRU)

is applied to map the question sequence $q = \{q_1, q_2, \ldots q_m\}$

into $h_q = \left(\overrightarrow{(h_q)_m}, \overleftarrow{(h_q)_m}\right)$

** GRU is computationally cheaper than LSTM.

All of the o/p states are used in paragraph representation but only the final states are used in question representation.

Attention:

In this step, goal is to compare the question embedding and all the contextual (para) embeddings, and select the pieces of information that are relevant to the question. Computation of a probability distribution α depending on the degree of relevance b/w word p_i and question q and then produce an o/p vector o which is a weighted combination of all contextual embeddings (p_i):

$$\alpha_i = \text{softmax}_i \; q^T W_s \tilde{p}_i = \text{softmax}(q^T W_s (H_s)_i)$$

$i = 1, \ldots, n$

$$o = \sum_i \alpha_i \tilde{p}_i$$

$W_s \in \mathbb{R}^{h \times h}$ is a bilinear term, which allows to compute a similarity b/w q and $\tilde{p}_i$ more flexibly than with just dot product;

Prediction:

Using the o/p vector o, the system outputs the most likely answer using:

$$a = \underset{a \in \text{vocab}}{\text{argmax}} \; W_a^T o$$

finally, the system adds a softmax f'n on top of $W a^T o$ and adopts a negative log-likelihood objective for training.