arXiv:1709.10204v1 [cs.CL] 29 Sep 2017

A Neural Comprehensive Ranker (NCR) for Open-Domain Question Answering

Bin Bi
Microsoft Research
One Microsoft Way
Redmond, WA 98052
bibi@microsoft.com

Hao Ma
Microsoft Research
One Microsoft Way
Redmond, WA 98052
haoma@microsoft.com

Abstract

This paper proposes a novel neural machine reading model for open-domain question answering at scale. Existing machine comprehension models typically assume that a short piece of relevant text containing answers is already identified and given to the models, from which the models are designed to extract answers. This assumption, however, is not realistic for building a large-scale open-domain question answering system which requires both deep text understanding and identifying relevant text from corpus simultaneously.

In this paper, we introduce Neural Comprehensive Ranker (NCR) that integrates both passage ranking and answer extraction in one single framework. A Q&A system based on this framework allows users to issue an open-domain question without needing to provide a piece of text that must contain the answer. Experiments show that the unified NCR is able to outperform the states-of-the-art in both retrieval of relevant text and answer extraction.

1 Introduction

An overview of our question answering system.

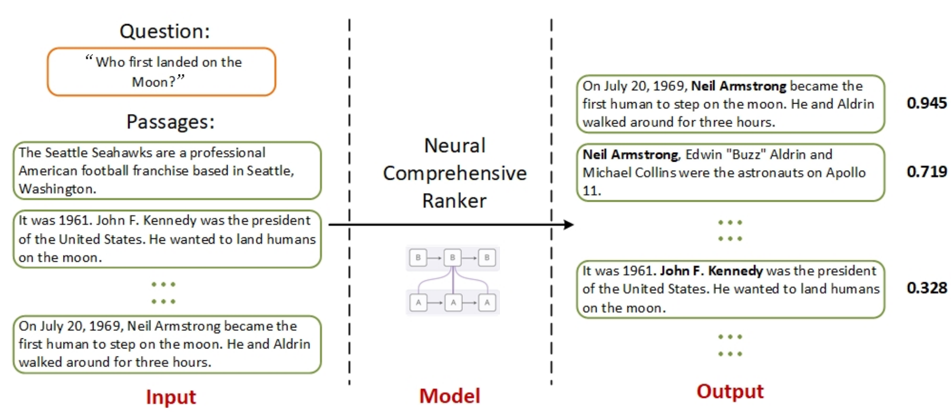

Figure 1: An Overview of Our Question Answering System

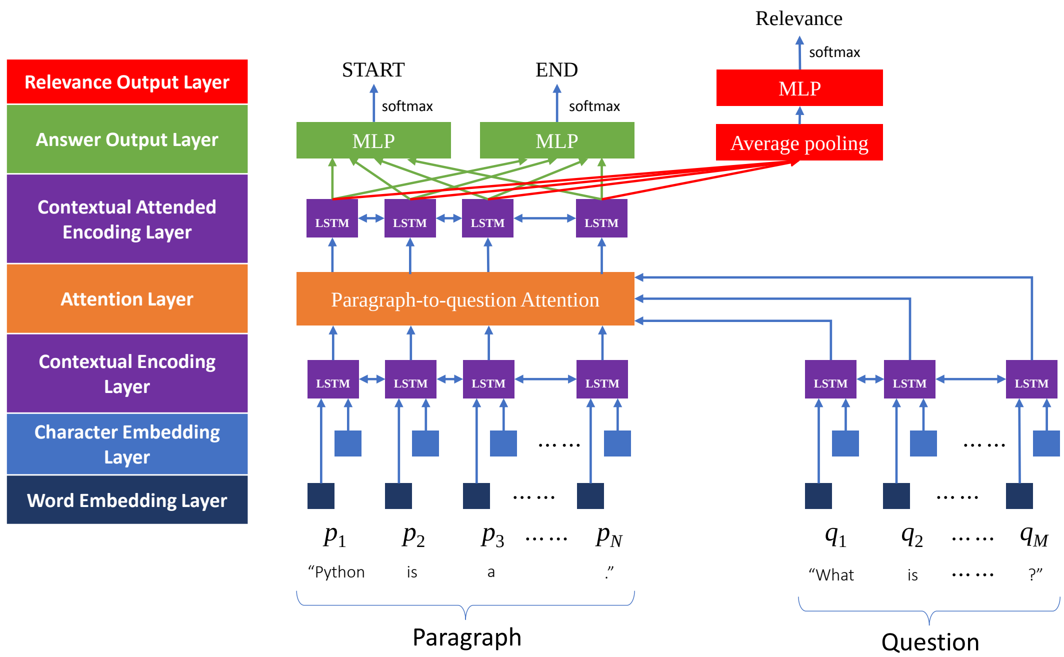

Figure 2: Neural Comprehensive Ranker (*best viewed in color*)

2 Neural Comprehensive Ranker

2.1 Model

The overall architecture of our Neural Comprehensive Ranker is illustrated in Figure 2. The model consists of seven layers:

1. **Word Embedding Layer** embeds each word into a fixed-length dense vector.
2. **Character Embedding Layer** maps each word into a fixed-length embedding using character-level CNNs [2].
3. **Contextual Encoding Layer** embeds each word with its surrounding context using bi-directional LSTM [1]. This layer enables the model to capture the sequential information of questions and paragraphs.
4. **Attention Layer** attends a paragraph to a given question, which produces a vector of question-aware features for each word in the paragraph.
5. **Contextual Attended Encoding Layer** models the sequential information of a paragraph at the attention level using another bi-directional LSTM, so that the context can be captured after attention.
6. **Answer Output Layer** produces the start and end indices of the answer span in a paragraph for a given question.
7. **Relevance Output Layer** produces the probability of a paragraph being relevant to a given question to which an answer can be found in this paragraph.

2.1.1 Word Embedding Layer

Let $q = \{v_1, v_2, \ldots, v_M\}$ and $p = \{x_1, x_2, \ldots, x_N\}$ denote the words in a given question q and a paragraph p, respectively. The word embedding layer encodes each word, v_i or x_i, into a high-dimensional embedding space. We use the pre-trained 100-dimensional GloVe word vectors [3] to initialize the word-level embedding.

2.1.2 Character Embedding Layer

The character embedding layer also embeds each word into a dense vector. In particular, we feed all characters of a word into a Convolutional Neural Network (CNN) with max-pooling to obtain a feature vector for the word.

Let $w = \{c_1, c_2, \dots, c_K\}$ denote the characters in a word w. The CNN has a set of filters $\{f(\cdot)\}$, where each filter $f(\cdot)$ reads in character embeddings in a word and derives a feature map $\mathbf{e}$:

$$\mathbf{e} = [e_1, e_2, \dots, e_{K-l+1}]^\top,$$

where each $e_i \in \mathbf{e}$ is given by

$$e_i = f(\mathbf{W_c} \cdot \mathbf{c}_{i:i+l-1} + b_c),$$

where $\mathbf{W_c}$ is a trainable parametric matrix, and b_c is a bias scalar. $\mathbf{c}_i$ is the embedding of the ith character in a word, and $\mathbf{c}_{i:i+l-1}$ denotes the concatenation of l character embeddings $\mathbf{c}_i, \mathbf{c}_{i+1}, \dots, \mathbf{c}_{i+l-1}$ in a sliding window of size l. A max-pooling then operates over $\mathbf{e}$ to generate one feature $\tilde{e}$ in the feature vector of the word:

$$\tilde{e} = f_{\texttt{MaxPool}}(\mathbf{e}) = \max_{e_i}\{e_i \in \mathbf{e}\}.$$

Why?

In our experiments, we set the sliding windows size l to 5 with 100 filters, which gives a 100-dimensional feature vector for each word.

We represent each word in a question and a paragraph as the concatenation of the character-level feature vector and word-level embedding, resulting in two matrices: $\mathbf{Q^a} \in \mathbb{R}^{L\times M}$ for a question and $\mathbf{P^a} \in \mathbb{R}^{L\times N}$ for a paragraph, where L is the embedding dimensionality, i.e., the sum of sizes of the feature vector and word embedding.

2.1.3 Contextual Encoding Layer

may also serve as encoder?

The contextual encoding layer aims to encode each word with its surrounding context to model word sequences in questions and paragraphs. To this end, we employ a Long Short-Term Memory (LSTM) on top of the embeddings produced by lower layers. At the ith time step, LSTM takes as input the embedding ($\mathbf{Q^a}_{i,:}$ or $\mathbf{P^a}_{i,:}$) of the current word, and computes a hidden state $\mathbf{h}_i$. We place an LSTM in both directions, and concatenate the outputs of the two LSTMs. For the ith word, hidden state $\mathbf{h}_i$ is semantically richer than embedding $\mathbf{Q^a}_{i,:}$ or $\mathbf{P^a}_{i,:}$, since $\mathbf{h}_i$ incorporates the contextual information from surrounding words to model the temporal interactions between words. This layer gives matrices of hidden states: $\mathbf{Q^b} \in \mathbb{R}^{J\times M}$ for a question, and $\mathbf{P^b} \in \mathbb{R}^{J\times N}$ for a paragraph, where J is the size of a hidden state.

3×4

2.1.4 Attention Layer

3×2 2×4

Given our goal of finding an answer from a paragraph relevant to a given question, the attention layer serves as a bridge to link the question to each paragraph. This layer takes as input contextual embeddings of both the question $\mathbf{Q^b}$ and the paragraph $\mathbf{P^b}$ produced by the contextual encoding layer, and outputs the question-aware representation of the paragraph $\mathbf{X} \in \mathbb{R}^{J\times N}$, where each column represents a paragraph word after attention.

# of words

This layer computes the affinity matrix $\mathbf{S} \in \mathbb{R}^{M\times N}$ containing an affinity score for each pair of a question word and a paragraph word. We calculate the affinity matrix as:

$$\mathbf{S}_{i,j} = \mathbf{w_a}^\top[\mathbf{Q^b}_{:,i}; \mathbf{P^b}_{:,j}; \mathbf{Q^b}_{:,i} \circ \mathbf{P^b}_{:,j}], \quad \text{Why?} \tag{1}$$

where $\mathbf{Q^b}_{:,i}$ is the ith column of matrix $\mathbf{Q^b}$, $\mathbf{P^b}_{:,j}$ is the jth column of matrix $\mathbf{P^b}$, $\mathbf{w_a} \in \mathbb{R}^{3J}$ is a trainable weight vector, $\circ$ is element-wise product, and $[;]$ is vector concatenation across row.

The affinity matrix is normalized to produce the attention weights $\mathbf{A} \in \mathbb{R}^{M\times N}$ across the question for each word in the paragraph:

$$\mathbf{A}_{:,j} = \text{softmax}(\mathbf{S}_{:,j}). \tag{2}$$

We then compute the attended question matrix in light of each paragraph word as:

$$\mathbf{X} = \underset{J\times M}{\mathbf{Q^b}}\,\underset{M\times N}{\mathbf{A}}. \quad J\times N \tag{3}$$

Each column of $\mathbf{X}$, corresponding to a paragraph word, indicates the sum of question vectors weighted by the relevance between individual question words and the paragraph word.

2.1.5 Contextual Attended Encoding Layer

$J \times N \to I \times N$

The contextual attended encoding layer is used to model sequential interactions in the context of a paragraph conditioned on a given question. It differs from the contextual encoding layer which encodes contextual information independent of the question at the level of word embedding. The contextual attended encoding layer takes as input the question-aware representation of the paragraph $\mathbf{X}$ from the attention layer, and produces a matrix $\mathbf{Z} \in \mathbb{R}^{I \times N}$, which is passed onto the next two output layers to predict relevance and an answer. In this layer, like the contextual encoding layer, we place another bi-directional LSTM, and concatenate the outputs from the two directions which gives the representation $\mathbf{Z}$. Each column of $\mathbf{Z}$ captures the surrounding context of the word at the attended level.

2.1.6 Answer Output Layer

Given the QA task of finding a sub-phrase of a relevant paragraph to answer a given question, the answer output layer predicts the start and the end indices of the answer span in the paragraph. To this end, we place two fully-connected layers on top of each column of $\mathbf{Z}$, and use softmax activation to obtain the probabilities of every word being the start and the end of the answer span, respectively:

$$p_{start} = \text{softmax}(\mathbf{w_s}^\top \mathbf{Z} + b_s), \tag{4}$$

$$p_{end} = \text{softmax}(\mathbf{w_e}^\top \mathbf{Z} + b_e), \tag{5}$$

where $\mathbf{w_s} \in \mathbb{R}^I$ and $\mathbf{w_e} \in \mathbb{R}^I$ are trainable weight vectors for the start and the end of the answer span, respectively. b_s and b_e are corresponding bias scalars.

2.1.7 Relevance Output Layer

The relevance output layer is used to identify whether a paragraph contains a sub-phrase that answers a given question. In this layer, we first place an average pooling layer to aggregate the word vectors $\mathbf{Z}$ produced by the contextual attended encoding layer into a single paragraph vector $\mathbf{v} \in \mathbb{R}^I$:

$$\mathbf{v} = f_{\texttt{AvgPool}}(\mathbf{Z}). \tag{6}$$

A fully-connected layer is then placed on top of the vector $\mathbf{v}$ of every paragraph to be evaluated:

$$\gamma = \mathbf{w_r}^\top \mathbf{v} + b_r, \tag{7}$$

where $\mathbf{w_r} \in \mathbb{R}^I$ is a trainable weight vector, and b_r is a bias scalar. Finally, we compute softmax activation on γs over all the paragraphs to be evaluated, which results in the probability that each of the paragraphs contains an answer to a given question.

2.2 Training

Given a QA training dataset D, our learning objective is to set the parameters θ of NCR so as to maximize the likelihood of all data samples from the training set. There are two types of data samples:

1. (q, p) pairs indicating paragraph p is relevant to question q to which an answer can thus be found in p.
2. a (START, END) pair for each (q, p) indicating the indices of start and end of the answer span in paragraph p for a given question q.

Let $p(r = 1|q, p; \theta)$ denote the probability that paragraph p is relevant to question q. Correspondingly, $p(r = 0|q, p; \theta) = 1 - p(r = 1|q, p)$ will be the probability of p being irrlevant to q. Also, we denote by $p(\text{START}|q, p; \theta)$ the probability that the answer span starts with the index START, and by $p(\text{END}|q, p; \theta)$ the probability of END being the end index of the answer span. As a result, the optimization objective is formally given by:

$$\begin{aligned} \boldsymbol{\theta}^* = &\arg\max_{\theta} \prod_{(q,p) \in D} p(r = 1|q, p; \theta) \prod_{(q,p) \notin D} p(r = 0|q, p; \theta) \\ &\times \prod_{(\text{START},\text{END}) \in D} p(\text{START}|q, p; \theta) p(\text{END}|q, p; \theta) \end{aligned} \tag{8}$$

where the first two multiplications reflect the likelihood of relevance and irrelevance, and the third multiplication reflects the likelihood of all answer spans.

As described before, we model the relevance probability by a sigmoid function: $p(r=1|q,p;\theta) = \sigma(p,q;\theta)$. To avoid a trivial solution to the maximization of the relevance likelihood, we need to present the model with irrelevant samples captured by $p(r=0|q,p;\theta) = 1-\sigma(p,q;\theta)$. The irrelevant samples are obtained by generating random (q, p) pairs, assuming they are all incorrectly mapped. The answer span probability, on the other hand, is modeled by a softmax function. The probabilities of all answer spans in the training dataset D are multiplied together as the answer span likelihood to be maximized.

The training loss for NCR is thus equivalently given by the negative sum of the log likelihoods over all data samples:

$$\begin{aligned} L(\theta) = &- \sum_{(q,p)\in D} \log(p(r=1|q,p;\theta)) - \sum_{(q,p)\notin D} \log(p(r=0|q,p;\theta)) \\ &- \sum_{(\text{START},\text{END})\in D} \log(p(\text{START}|q,p;\theta)) + \log(p(\text{END}|q,p;\theta)) \end{aligned} \tag{9}$$

Training the NCR model is achieved by finding the parameters θ so as to minimize the loss function given in Equation (9).

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[2] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751, 2014.

[3] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.