Length-adaptive Neural Network for Answer Selection

Taihua Shao
Science and Technology on Information Systems Engineering Laboratory
National University of Defense Technology
Changsha, China
shaotaihua13@nudt.edu.cn

Fei Cai*
Science and Technology on Information Systems Engineering Laboratory
National University of Defense Technology
Changsha, China
caifei@nudt.edu.cn

Honghui Chen
Science and Technology on Information Systems Engineering Laboratory
National University of Defense Technology
Changsha, China
chh0808@gmail.com

Maarten de Rijke
Informatics Institute
University of Amsterdam
Amsterdam, The Netherlands
derijke@uva.nl

ABSTRACT

Answer selection focuses on selecting the correct answer for a question. Most previous work on answer selection achieves good performance by employing an RNN, which processes all question and answer sentences with the same feature extractor regardless of the sentence length. These methods often encounter the problem of long-term dependencies. To address this issue, we propose a Length-adaptive Neural Network (LaNN) for answer selection that can auto-select a neural feature extractor according to the length of the input sentence. In particular, we propose a flexible neural structure that applies a BiLSTM-based feature extractor for short sentences and a Transformer-based feature extractor for long sentences. To the best of our knowledge, LaNN is the first neural network structure that can auto-select the feature extraction mechanism based on the input. We quantify the improvements of LaNN against several competitive baselines on the public WikiQA dataset, showing significant improvements over the state-of-the-art.

CCS CONCEPTS

• **Information systems → Question answering**.

KEYWORDS

Answer selection, Neural network, Question answering.

ACM Reference Format:
Taihua Shao, Fei Cai, Honghui Chen, and Maarten de Rijke. 2019. Length-adaptive Neural Network for Answer Selection. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3331184.3331277

*Corresponding author.

SIGIR '19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331277

1 INTRODUCTION

Answer selection plays a crucial role in a Question Answering (QA) system, which selects the most appropriate answer for a question from a set of candidate answers. Deep learning-based approaches have been well studied for this task [4]. These approaches mainly attempt to generate high-quality sentence embeddings of question and answer, which are then utilized to measure the relevance of a candidate answer to a question.

Previous research has achieved impressive performance on this task. For instance, Tan et al. [6] employ an attentive BiLSTM to measure the relevance of segments in candidate answers for a particular question. Wang and Nyberg [8] apply a stacked BiLSTM-based feature extractor to learn a joint feature vector for question-answer pairs. However, these RNN-based models do not consider an important aspect that has been shown to affect sentence embedding-based tasks, namely the length of a sentence. As the sentence length grows, models will suffer from a long-term dependency problem due to the sequential nature of RNNs. It becomes difficult to learn dependencies between words in distant positions. This, in turn, impacts the quality of embeddings for long sentences, as long-distance interactions between words of may be not be captured [3].

To address this issue, we propose a *Length-adaptive Neural Network* (LaNN) for the task of answer selection. Specifically, we deploy a hierarchical length-adaptive neural structure to generate sentence embeddings for question and answer sentences, which aims at extracting high-quality sentence features by employing a different neural feature extractor depending on the length of the input sentences. We first generate a word representation for each word in the input sentence by concatenating a *frozen* word embeddingand a *fine-tuning* word embedding. Then, we propose a flexible neural structure that applies a BiLSTM-based feature extractor for short sentences and a Transformer-based feature extractor for long sentences, respectively. Finally, an attentive pooling layer that takes the interaction between question and answer sentences into consideration is employed to generate the sentence embeddings that are used to measure the relevance of a question and an answer.

We evaluate the performance of the proposed LaNN model against several competitive baselines on a popular QA dataset,

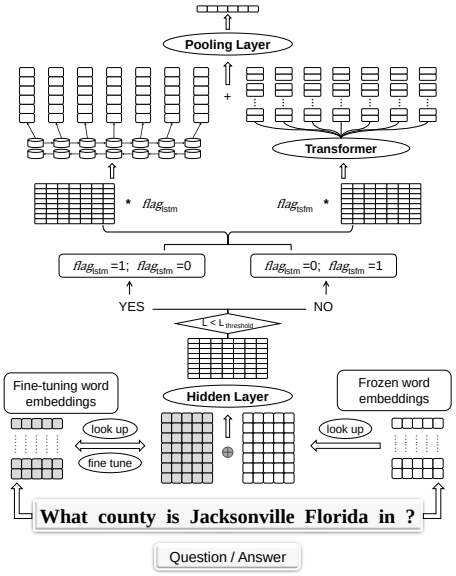


Figure 1: Overview of the length-adaptive neural network for answer selection.

WikiQA. The experimental results show that the LaNN model outperforms the state-of-the-art baselines, showing a general improvement of 1.67% and 2.06% in terms of MAP and MRR, respectively.

Our main technical contributions are: (1) The LaNN model for answer selection, which can auto-select a neural feature extractor for questions and answers based on the length of an input sentence; and (2) An evaluation of the performance of LaNN, which shows that it improves performance on the answer selection task, especially for short questions with long correct answers.

2 APPROACH

Fig. 1 presents an overview of the proposed length-adaptive neural network for answer selection. The model consists of three main components, i.e., word representation (see §2.1), feature extractor (see §2.2), and sentence embedding and answer ranking (see §2.3).

2.1 Word representation

Let s be an input sentence (question or answer) with length L. To keep as many word features as possible, we produce the word representation corresponding to each word w_t in s not only from frozen pre-trained word embeddings but fine-tuning word embeddings. Then, we concatenate two word embeddings to form a combined word vector. Finally, we deploy a hidden layer to select useful features from the concatenated word vector. The final representation r_{w_t} (of dimension D) of the word w_t is calculated according to:

$$r_{w_t} = \tanh(W_h \cdot (r^{fr}_{w_t} \oplus r^{ft}_{w_t}) + b_h), \tag{1}$$

where $r^{fr}_{w_t}$ and $r^{ft}_{w_t}$ are corresponding word vectors from the frozen word embeddings and the fine-tuning word embeddings; $W_h \in \mathbb{R}^{D\times D}$ and $b_h \in \mathbb{R}^{D\times 1}$ are network parameters of the hidden layer. The word representations of s form the word representation matrix:

$$R_W = \begin{pmatrix} r_{w_1} & r_{w_2} & \cdots & r_{w_L} \end{pmatrix}. \tag{2}$$

2.2 Feature extractor

We design a length-adaptive neural network as the feature extractor, making good use of the context information of words in s to generate a high-quality sentence representation for s. Most previous neural networks for answer selection [e.g., 1, 6] do not distinguish between input sentences of different lengths. In contrast, we deploy a BiLSTM-based and a Transformer-based feature extractor to deal with sentences of different lengths for generating sentence embeddings. We employ two flags, i.e., $flag_{\text{lstm}}$ and $flag_{\text{tsfm}}$, for each feature extractor according to the sentence length L as follows:

$$\begin{cases} flag_{\text{lstm}} = 1 \text{ and } flag_{\text{tsfm}} = 0, (L < L_{\text{threshold}}) \\ flag_{\text{lstm}} = 0 \text{ and } flag_{\text{tsfm}} = 1, (L \geq L_{\text{threshold}}), \end{cases} \tag{3}$$

where $L_{\text{threshold}}$ is a preset threshold to judge whether the input sentence s is long or not. The flags will be employed to activate the corresponding feature extractor by multiplying the input word representation matrix R_W with the value of flags as follows:

$$R^{\text{lstm}}_W = R_W \cdot flag_{\text{lstm}} \tag{4}$$

$$R^{\text{tsfm}}_W = R_W \cdot flag_{\text{tsfm}}, \tag{5}$$

where R^{lstm}_W and R^{tsfm}_W are updated input representation matrixes for the BiLSTM-based and Transformer-based feature extractors.

For a short sentence, we activate the BiLSTM-based feature extractor by setting $flag_{\text{lstm}}$ to 1 and $flag_{\text{tsfm}}$ to 0, which leads to a null representation matrix in the Transformer-based feature extractor. Operations on the t-th word representation $r^{\text{lstm}}_{w_t}$ in R^{lstm}_W are:

$$\begin{cases} f_t &= \sigma(W_f \cdot r^{\text{lstm}}_{w_t} + U_f \cdot h_{t-1} + b_f) \\ i_t &= \sigma(W_i \cdot r^{\text{lstm}}_{w_t} + U_i \cdot h_{t-1} + b_i) \\ \widetilde{C_t} &= \tanh(W_c \cdot r^{\text{lstm}}_{w_t} + U_c \cdot h_{t-1} + b_c) \\ C_t &= f_t \cdot C_{t-1} + i_t \cdot \widetilde{C_t} \\ o_t &= \sigma(W_o \cdot r^{\text{lstm}}_{w_t} + U_o \cdot h_{t-1} + b_o) \\ h_t &= o_t \cdot \tanh(C_t), \end{cases} \tag{6}$$

where i, f and o represent the input gate, the forget gate and the output gate, respectively; h represents the memorized word representation (of dimension H); $\widetilde{C}$ and C are the overall and the present memory; σ is the activation function sigmoid; $W \in \mathbb{R}^{H\times D}$, $U \in \mathbb{R}^{H\times H}$ and $b \in \mathbb{R}^{H\times 1}$ are the network parameters, determining the input information, output information and bias, respectively. After deploying an LSTM in two directions, we obtain a BiLSTM-based sentence representation matrix R^{LS}_s as follows:

$$r^{LS}_{w_t} = \overrightarrow{h_t} \oplus \overleftarrow{h_t}, \tag{7}$$

$$R^{LS}_s = \begin{pmatrix} r^{LS}_{w_1} & r^{LS}_{w_2} & \cdots & r^{LS}_{w_L} \end{pmatrix}, \tag{8}$$

where || indicates concatenation of two vectors.

For a long sentence, we activate the Transformer-based feature extractor by setting $flag_{\text{lstm}}$ to 0 and $flag_{\text{tsfm}}$ to 1. Following [7], we employ a positional encoding to inject sequential information and generate an updated R^{tsfm}_W. After that, a scaled perception function is applied to calculate self-attentive similarity in R^{tsfm}_W:

$$f(R^{\text{tsfm}}_W, R^{\text{tsfm}}_W) = O^{\text{T}}_a \cdot \tanh(W_a \cdot R^{\text{tsfm}}_W + U_a \cdot R^{\text{tsfm}}_W), \tag{9}$$

where $O_a \in \mathbb{R}^{D\times L}$, $W_a \in \mathbb{R}^{D\times D}$ and $U_a \in \mathbb{R}^{D\times D}$ are the attention parameters. Then, the self-attentive sentence representation matrix

R_s^a is produced by:

$$R_s^a = R_W^{\text{tsfm}} \cdot \text{softmax}\left(\frac{f(R_W^{\text{tsfm}}, R_W^{\text{tsfm}})^{\text{T}}}{\sqrt{d_{R_W^{\text{tsfm}}}}}\right) = \begin{pmatrix} r_{w_1}^a & r_{w_2}^a & \cdots & r_{w_L}^a \end{pmatrix}. \quad (10)$$

where $d_{R_W^{\text{tsfm}}}$ is the dimension of the row vector in R_W^{tsfm}; $\sqrt{d_{R_W^{\text{tsfm}}}}$ aims to scale the softmax function to regulate its value. Finally, $r_{w_t}^a$ is the t-th self-attentive word vector in R_s^a.

We then adopt a multi-head mechanism [7] to jointly incorporate information from different representation subspaces. Assuming that there are n heads in the Transformer-based feature extractor, the n self-attentive sentence representation matrices will be concatenated to form the output sentence representation matrix R_s^{TF} as:

$$R_s^{TF} = R_s^{a_1} \oplus R_s^{a_2} \oplus \cdots \oplus R_s^{a_n} = \begin{pmatrix} r_{w_1}^{TF} & r_{w_2}^{TF} & \cdots & r_{w_l}^{TF} \end{pmatrix}, \quad (11)$$

where $R_s^{a_i}$ is the i-th self-attentive sentence representation matrix.

2.3 Sentence embedding and answer ranking

After generating the sentence representation matrix in §2.2, we employ an attentive pooling to generate the sentence embeddings v_q and v_a for question and answer from their corresponding sentence representation matrixes R_Q and R_A according to:

$$G = {R_Q}^{\text{T}} \cdot U \cdot R_A, \quad (12)$$

$$v_q = R_Q \cdot \text{softmax}(ColumnMax(G)), \quad (13)$$

$$v_a = R_A \cdot \text{softmax}(RowMax(G)), \quad (14)$$

where G is the attentive similarity between R_Q and R_A; $U \in \mathbb{R}^{D \times D}$ is the attention parameter; $ColumnMax(\cdot)$ (or $RowMax(\cdot)$) is a function that returns the max value of a column (or row) vector of a matrix. The relevance of an answer to a question is computed using the cosine similarity of the sentence embeddings [5, 6].

In the training phase, each training instance consists of a question q, a positive answer a^+ (a ground truth) and a negative answer a^- (an incorrect answer) randomly sampled from all answers in the training set. We train the neural network for the best training epoch by minimizing the following ranking loss of candidate answers:

$$loss = \max\{0, m - \cos(v_q, v_{a^+}) + \cos(v_q, v_{a^-})\}, \quad (15)$$

where m is a preset margin to judge if a training instance will be terminated or not. By doing so, we can rank the candidate answers according to their relevance towards the corresponding question.

3 EXPERIMENTS

Model summaries. We examine the effectiveness of the proposed LaNN model by comparing its performance against the following competitive state-of-the-art baselines: (1) QA-CNN [1]: a CNN-based model that employs a CNN-based feature extractor behind a hidden layer to generate sentence embeddings. (2) QA-BiLSTM [6]: a BiLSTM-based model that employs the BiLSTM-based feature extractor to generate sentence embeddings. (3) AB-LSTM/CNN [6]: an attention-based hybrid model that applies a serial structure to combine CNN and BiLSTM to generate sentence embeddings. LaNN is the answer selection model proposed in this paper.

Table 1: WikiQA corpus statistics.

Variables	Training	Validation	Test	Overall
#Questions	873	126	243	1,242
#Correct Answers	1,040	140	293	1,473
QA Pairs	8,672	1,130	2,351	12,153
Avg. len. of ques.	6.36	6.72	6.46	6.42
Avg. len. of answ.	25.51	24.59	25.02	25.33

Table 2: Main experimental settings. Bs: batch size; Mg: margin; Dp: dropout; Lr: learning rate; L2: L2 regularization coefficient; Hn and Hs: the number and the size of the Transformer head; Rs: hidden size of BiLSTM; Dr: decay rate.

Model	Bs	Mg	Dp	Lr	L2	Hn	Hs	Rs	Dr
LaNN	30	0.1	0.5	10^{-4}	10^{-3}	7	40	280	0.85

Research questions. (RQ1) Does LaNN beat competitive answer selection models? **(RQ2)** How does LaNN compare to baseline models for question-answer pairs of different lengths, i.e., short questions with long answers (*short-long*) and long questions with long answers (*long-long*)? (As all answers in WikiQA are long, *short-short* and *long-short* question-answer pairs are absent.)

Dataset and parameters. The dataset we use to evaluate the performance of LaNN is a publicly available open domain dataset, the WikiQA dataset released in 2015 [9]; statistics of the WikiQA dataset are listed in Table 1.

We set the length threshold to 5 in our experiments,[1] which is close to the average length of questions. Following [5], the pre-trained word embedding's dimension and the size of the hidden layer are set to 300. We pad the sentence length for all questions and answers to 40 [5]. For optimizing the loss, Adam [2] is employed. We train our models in mini-batches and employ exponential decay to vary the learning rate in every epoch. *L2* regularization and dropout methods are included in our training process to avoid over-fitting. Table 2 details the main parameters of LaNN.

Evaluation metrics. We view the answer selection task as a ranking problem; it is aimed at ranking the candidate answers according to their relevance towards the question. Hence, following prior research on answer selection [8], we adopt Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) as evaluation metrics.

4 RESULTS AND DISCUSSION

4.1 Overall performance

To answer **RQ1**, we present the results of the three baselines and the LaNN model on the test set of WikiQA. In addition, we investigate the model performance on questions of various types. In particular, we categorize the test set into 5 groups according to the question type, i.e., *how*, *what*, *who*, *where*, and *when*. The detailed evaluation scores in terms of MAP and MRR are presented in Table 3.

As shown in Table 3, in general, QA-BiLSTM beats QA-CNN in terms of both metrics. AB-LSTM/CNN shows superiority compared

[1] We test different threshold values in preliminary experiments but the best performance was observed with a value of 5.

Table 3: Model performance in terms of MAP and MRR. For each category, the best result is highlighted.

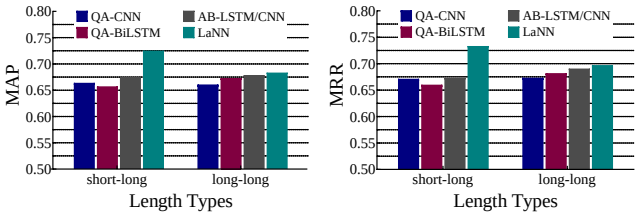
Models	MAP						MRR					
	overall	how	what	who	where	when	overall	how	what	who	where	when
QA-CNN	0.6613	0.5837	0.6887	0.6439	0.6845	0.6169	0.6732	0.5919	0.7044	0.6493	0.6932	0.6221
QA-BiLSTM	0.6709	**0.6061**	0.6812	0.7426	0.7121	0.5259	0.6790	**0.6115**	0.6945	0.7450	0.7080	0.5259
AB-LSTM/CNN	0.6780	0.5764	0.6875	0.7561	0.7265	0.6008	0.6882	0.5802	0.7003	0.7629	0.7376	0.6094
LaNN	**0.6893**	0.5621	**0.6933**	**0.8005**	**0.7576**	**0.6196**	**0.7024**	0.5721	**0.7083**	**0.8083**	**0.7761**	**0.6273**

to the other two baselines without integrating an attention mechanism. Regarding LaNN, compared to the baselines, it achieves the highest performance in terms of MAP and MRR. In particular, LaNN outperforms QA-BiLSTM with up to 2.74% and 3.45% in terms of MAP and MRR, respectively. The Transformer-based feature extractor in LaNN can help deal with the long-term dependencies problem in the long sentences. Furthermore, the overall MAP and MRR scores of LaNN are increased by up to 1.67% and 2.06% against the best baseline AB-LSTM/CNN. This indicates an LaNN model can help improve the performance of answer selection.

As to questions of different types, LaNN beats three baselines for questions of all types except for type *how*. The baseline model with the BiLSTM structure, QA-BiLSTM, presents the best results; the BiLSTM-based structure is more effective than the CNN structure as well as the Transformer structure in extracting contextual features hidden in sequential data that is prominent in answering the questions with the type *how*. The LaNN model achieves its highest improvements in the *who* group, with an increase of 5.87% and 5.95% against the best baseline AB-LSTM/CNN in terms of MAP and MRR, respectively. The Transformer-based structure employed in the LaNN model, which can deal with long-term dependencies to some extent, is good at extracting features hidden in long-range words for answering the question with the type *who*.

4.2 Impact of length type

To answer **RQ2**, we consider *short-long* and *long-long* question answer pairs; as explained previously, these are the only pairs available in the WikiQA dataset. We plot the results in Fig. 2. Compared



(a) MAP. **(b) MRR.**

Figure 2: Performance on different length types of question-answer pairs in terms of MAP and MRR.

to the baselines, LaNN shows obvious improvements in terms of MAP and MRR for the *short-long* group, while the improvements for the *long-long* are modest. For instance, LaNN outperforms the best baseline AB-LSTM/CNN by 7.34% and 8.99% in terms of MAP and MRR on the *short-long* group; the increases in MAP and MRR are only 0.74% and 0.97% on the *long-long* group.

Thus, LaNN can help improve the performance of answer selection, especially for question-answer pairs with short question and long correct answers. That is, the Transformer-based feature extractor is beneficial for long answers; but when dealing with long answers it is more beneficial for short questions than for long ones. We further analyze the impact of length gap between question and answer: the longer the answer is and the shorter the question is, the better performance our LaNN model achieves.

5 CONCLUSIONS AND FUTURE WORK

We propose a length-adaptive neural network (LaNN) for answer selection, which employs a BiLSTM-based feature extractor and a Transformer-based feature extractor to capture global interactions of words to obtain improved sentence embeddings. LaNN can auto-select the neural feature extractor according to the length of the input sentence. Experimental results show LaNN can achieve considerable improvements in terms of MAP and MRR over state-of-the-art baselines. Applying different neural feature extractors to short questions and long answers leads to substantial improvements in of answer selection performance. As to future work, we would like to examine the scalability of our proposal by evaluating its effectiveness on other datasets. In addition, we have interest in applying the proposed length-adaptive neural network to other tasks, e.g., text summarization and natural language inference.

ACKNOWLEDGMENTS

This research was supported by the National Natural Science Foundation of China under No. 61702526, the Defense Industrial Technology Development Program under No. JCKY2017204B064, Ahold Delhaize, the Association of Universities in the Netherlands (VSNU), and the Innovation Center for Artificial Intelligence (ICAI).

REFERENCES

[1] Minwei Feng, Bing Xiang, Michael R Glass, et al. 2015. Applying deep learning to answer selection: A study and an open task. In *ASRU'2015*. 813–820.
[2] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[3] Xiangpeng Li, Jingkuan Song, et al. 2019. Beyond RNNs: Positional Self-Attention with Co-Attention for Video Question Answering. In *AAAI'19*. To appear.
[4] Jinfeng Rao, Hua He, and Jimmy Lin. 2017. Experiments with convolutional neural network models for answer selection. In *SIGIR'17*. ACM, 1217–1220.
[5] Cicero Dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive Pooling Networks. *arXiv: Computation and Language* (2016).
[6] Ming Tan, Cicero Dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *ACL'16*. 464–473.
[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, et al. 2017. Attention is all you need. In *NIPS'17*. 5998–6008.
[8] Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *ACL'15*. 707–712.
[9] Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *EMNLP'15*. 2013–2018.