S^3-NET: SRU-Based Sentence and Self-Matching Networks for Machine Reading Comprehension

CHEONEUM PARK, Kangwon National University
HEEJUN SONG, Samsung Research
CHANGKI LEE, Kangwon National University

Machine reading comprehension question answering (MRC-QA) is the task of understanding the context of a given passage to find a correct answer within it. A passage is composed of several sentences; therefore, the length of the input sentence becomes longer, leading to diminished performance. In this article, we propose S^3-NET, which adds sentence-based encoding to solve this problem. S^3-NET, which is based on a simple recurrent unit architecture, is a deep learning model that solves the MRC-QA by applying matching network to sentence-level encoding. In addition, S^3-NET utilizes self-matching networks to compute attention weight for its own recurrent neural network sequences. We perform MRC-QA for the SQuAD dataset of English and MindsMRC dataset of Korean. The experimental results show that for SQuAD, the S^3-NET model proposed in this article produces 71.91% and 74.12% exact match and 81.02% and 82.34% F1 in single and ensemble models, respectively, and for MindsMRC, our model achieves 69.43% and 71.28% exact match and 81.53% and 82.77% F1 in single and ensemble models, respectively.

CCS Concepts: • **Information systems** → **Question answering**; • **Computing methodologies** → Information extraction;

Additional Key Words and Phrases: Machine reading comprehension, question answering, Korean MRC-QA, sentence and self-matching network, hierarchical model, simple recurrent unit

ACM Reference format:
Cheoneum Park, Heejun Song, and Changki Lee. 2020. S^3-NET: SRU-Based Sentence and Self-Matching Networks for Machine Reading Comprehension. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 19, 3, Article 35 (February 2020), 14 pages.
https://doi.org/10.1145/3365679

1 INTRODUCTION

Machine reading comprehension question answering (MRC-QA) addresses problems such as the CBT [7] of the Facebook bAbi task; WikiQA [2], which seeks sentences with correct answers in a given context; and Stanford's SQuAD [18], which determines the start and end boundaries of

This work was supported by Institute for Information & Communications Technology Promotion (IITP) grants funded by the Korean government (MSIT) (2018-0-00605, Artificial Intelligence Contact Center Solution; 2013-0-00131, Development of Knowledge Evolutionary WiseQA Platform Technology for Human Knowledge Augmented Services).
Authors' addresses: C. Park and C. Lee, Kangwon National University, 1 Gangwondaehakgil, Chuncheon, Gangwondo, 21341, KOR; emails: {parkce, leeck}@kangwon.ac.kr; H. Song, Samsung Research, Samsung Electronics Co., Ltd. 56, Seongchon-gil, Seoul, 06765, KOR; email: heejun7.song@samsung.com.

2375-4699/2020/02-ART35 $15.00
https://doi.org/10.1145/3365679

a correct answer in a given context. Moreover, this method can also be applied in more complex domains, such as Microsoft's MS-MARCO [13].

To complete such tasks, the machine must first understand a given context and find the correct answer in the context. This understanding is called *machine reading comprehension* (MRC). The following is an example using the SQuAD-MRC system: for the question "What list was Warsaw's Old Town inscribed onto in 1980?," the system analyzes the relevant passage ("After World War II, under a Communist regime set up . . . form. In 1980, Warsaw's historic Old Town was inscribed onto UNESCO's World Heritage list.") and finds the right answer ("UNESCO's World Heritage list").

End-to-end deep learning models such as DrQA [2], FastQA [23], r-net [22], and Bidirectional Attention Flow (BiDAF) [19] have also been studied with regard to performing MRC-QA tasks. These models perform encoding and matching for given questions and passages, and output the boundary indices of correct answers (i.e., the start and end position of the correct answer). Moreover, these models use pointer networks [21] based on attention mechanism [1]. Since a single paragraph given in the SQuAD dataset is composed of several sentences, the length of the paragraph becomes longer, making MRC increasingly difficult. In this article, we propose a model (S^3-NET) that attempts to solve this problem by sentence unit encoding to better understand the meaning of each sentence. S^3-NET has a hierarchical structure that performs sentence-level encoding based on simple recurrent unit architecture (SRU) [12], applies a matching layer, and uses a self-matching network [22].

In this article, we propose S^3-NET with a sentence and self-matching network to an encoder recurrent neural network (RNN) using multi-layer SRU. To summarize, the main contributions of our model are as follows:

(1) *Model performance enhancement*: S^3-NET performs a feature extraction on question and passage input sequence and a self-matching network that computes alignment weights between words with similar meanings.
(2) *Hierarchical model*: S^3-NET has a hierarchical structure between a word sequence of the passage and a sentence sequence of a passage. When calculating the answer score of a passage in the output layer, the passage representation and the sentence representation are calculated together so that both representations can be learned together.
(3) *SRU for speed improvement*: We apply the SRU to improve the training and testing operation speed of the S^3-NET, a more complex model than DrQA or fastQA models.
(4) *Korean MRC-QA dataset*: We applied the proposed model to the Korean MindsMRC dataset [15] constructed in the SQuAD style as well as the SQuAD dataset, which is an English MRC-QA dataset, and showed the best performance in the MindsMRC dataset.

2 SIMPLE RECURRENT UNIT

Although similar to gated recurrent unit architecture (GRU) [4], SRU solves the vanishing gradient problem that occurs when back-propagation of the RNN is performed by using a neural gate and long short-term memory (LSTM) [5]. SRU is a recurrent unit model that improves speed by removing previous hidden states from the gate input. It uses input gate i_t, forget gate f_t, and a highway network. The equations are as follows:

$$\tilde{x}_t = Wx_t, \tag{1}$$

$$i_t = (1 - f_t), \tag{2}$$

$$f_t = \sigma(W_f x_t + b_t), \tag{3}$$

$$r_t = \sigma(W_r xt + br), \tag{4}$$

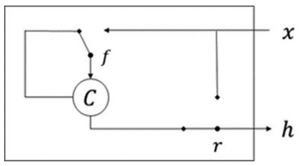


Fig. 1. Simple recurrent unit.

$$c_t = f_t \odot c + t - 1 + i_t \odot \tilde{x}_t, \tag{5}$$

$$h_t = r_t \odot g(c_t) + (1 - r_t) \odot x_t. \tag{6}$$

Input gate i_t performs as element-wise multiplication with $\tilde{x}_t$ to determine whether to reflect the input information, and forget gate f_t performs an element-wise multiplication with c_{t-1} to determine how much of the previous internal state information should be reflected. $\tilde{x}_t$ is the result of linear transformation obtained by multiplying the input x_t by the weight W_f, f_t is the result of applying a feed-forward neural network (FFNN) on input x_t with a sigmoid, and i_t is equal to $i_t = 1 - f_t$. The forget gate of existing RNN models such as GRU and LSTM includes the previous hidden state h_{t-1} as $f_t = \sigma(W_r x_t + R h_{t-1} + b_t)$. The SRU system applies an FFNN to the gates to reduce the amount of computation and to enable parallel computation. c_t controls the information transfer from the input x_t and the previous internal state c_{t-1} to the new internal state, and it applies the activation function $g(\cdot)$ to produce the output of that state. The hidden state h_t is the result of executing the highway network for the internal state output and the input x_t. Here, the output $g(c_t)$ determines how much the internal state output is reflected to the hidden state by element-wise multiplication with the reset gate r_t. The input x_t is calculated by $(1 - r_t)$ and element-wise multiplication to determine whether to reflect the input x_t. The SRU system is shown in Figure 1.

3 PROPOSED MODEL

To perform MRC, each model is given a dataset of questions (Q), passages (P), and correct answers (Y). Questions are m words, where $Q = \{q_1, q_2, \ldots, q_m\}$. Passages consists of n words, where $P = \{p_1, p_2, \ldots, p_n\}$. To perform sentence encoding, the input word sequence P' (composed of sentences) is used, where $P' = \{p'_{1,1}, p'_{2,1}, \ldots, p'_{t_n,t}\}$, t is the sentence index, and t_n is the word index for each sentence. The MRC system encodes Q, P, and P' using a hierarchical model, and outputs start boundary $y_1(P_{start})$ and end boundary $y_2(P_{end})$ of the answer using pointer networks.

In this article, we propose hierarchical structure-based S^3-NET coupled with a passage, question, and sentence-level encoder based on SRU. We use question-sentence-matching, sentence-passage-matching and self-matching networks. The S^3-NET model is shown in Figure 2.

S^3-NET learns feature embedding for passages and questions in the feature layer and concatenates them to produce $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{q}}$. Thereafter, passage encoding and question encoding are performed using bidirectional SRU (BiSRU) in each hidden layer. In this article, we used similar features to DrQA such as the word embedding, character embedding, exact match (EM), token feature, and aligned sentence embedding as follows:

- *Exact match*: The EM feature is a binary feature (1 or 0), which verifies that word p_i in the paragraph is included in questions for original, lowercase, or lemma form:

$$f_{exact\_match}(p_i) = \mathbb{I}(p_i \in Q). \tag{7}$$

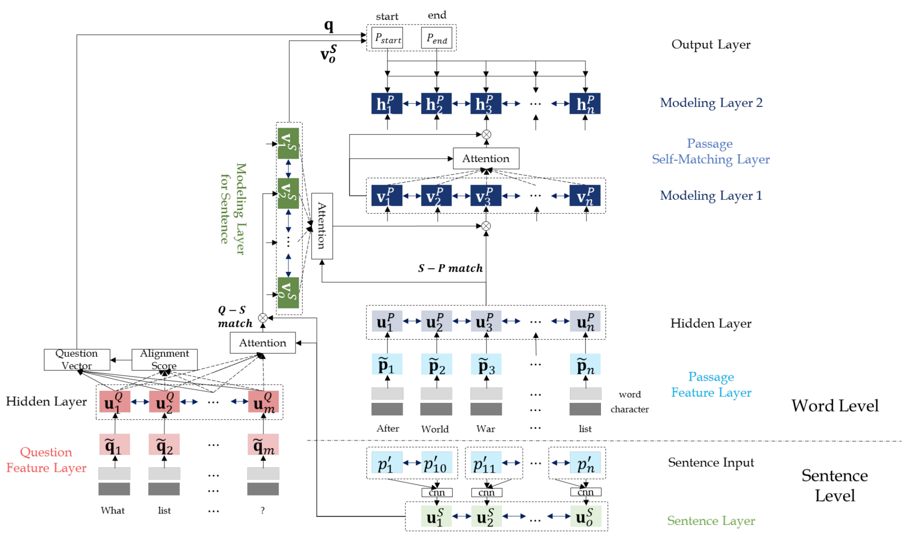


Fig. 2. S^3-NET model structure.

- *Token feature*: The token feature normalizes the term frequency $tf(p_i)$ (T is a length of an each sequence for question or passage):

$$f_{tf}(p_i) = tf(p_i)/T. \tag{8}$$

- *Aligned sentence embedding*:

$$f_{align}(p_i) = \sum_j a_{i,j} E(q_j), \tag{9}$$

$$a_{i,j} = \frac{\exp(\alpha(E(p_i)) \cdot \alpha(E(q_j)))}{\sum_{j'} \exp(\alpha(E(p_i)) \cdot \alpha(E(q_{j'})))}. \tag{10}$$

  The aligned sentence embedding function computes a matching context vector, which is then multiplied by both the context encoder vector and alignment score for passage and question representation. We use single dense layer $\alpha(.)$ with ReLU nonlinearity. The layer is applied to the output of the word embedding layer. $E(.)$ is the embedding layer to map word to embedding representation.
- *POS and NER tag*: Our proposed system uses 51-dimensional part-of-speech (POS) tag embedding for 51 distinct types of the POS tags and 18-dimensional named entity recognition (NER) embedding for 18 distinct types of the NER tags.

We apply the preceding features (the EM, token feature, and aligned sentence embedding) to both the passages and the questions.

The equations for the passage encoding $\mathbf{u}_t^P$ and question encoding $\mathbf{u}_t^Q$ are as follows:

$$\mathbf{u}_t^P = BiSRU_P\left(\mathbf{u}_{t-1}^P, \tilde{\mathbf{p}}_t\right), \tag{11}$$

$$\mathbf{u}_t^Q = BiSRU_Q\left(\mathbf{u}_{t-1}^Q, \tilde{\mathbf{q}}_t\right). \tag{12}$$

The question performs the encoding and then creates an alignment vector b_j using an RNN, and then calculates b_j with the encoding hidden state of the question to produce question vector

q. The expression for the question vector **q** is used to output the correct answer in output layer. **w** is weight vector to learn:

$$\mathbf{q} = \sum_j b_j \mathbf{u}_j^Q, \tag{13}$$

$$b_j = \frac{\exp(\mathbf{w} \cdot \mathbf{u}_j^Q)}{\sum_{j'} \exp(\mathbf{w} \cdot \mathbf{u}_{j'}^Q)}. \tag{14}$$

When constructing the sentence encoding, only the word embedding of the input word is considered, and the convolutional neural network (CNN) [9] is applied to each sentence to generate the hidden state $\mathbf{p}^S$. RNN is performed as shown to create the sentence encoding as follows:

$$\mathbf{u}_t^S = BiSRU_S\left(\mathbf{u}_{t-1}^S, \mathbf{p}_t^S\right). \tag{15}$$

For the question-sentence ($Q - S$ *match*) matching layer, the question encoding $\mathbf{u}_m^Q$ is included in the sentence encoding $\mathbf{u}_o^S$ by using a gated attention-based RNN. Next, this is modeled as a sentence encoding vector in the modeling layer for sentence to create $\mathbf{v}_o^S$, which is expressed as follows:

$$\mathbf{v}_t^S = BiSRU_{QS}\left(\mathbf{v}_{t-1}^S, \mathbf{g}_t \odot \left[\mathbf{u}_t^S; \mathbf{c}_t\right]\right), \tag{16}$$

$$\mathbf{g}_t = sigmoid\left(\mathbf{W}_g^S\left[\mathbf{u}_t^S; \mathbf{c}_t\right]\right), \tag{17}$$

$$\alpha_i^t = \frac{\exp(\mathbf{u}_t^S \mathbf{W}_\alpha^S \mathbf{u}_i^Q)}{\sum_{j=1}^M \exp(\mathbf{u}_t^S \mathbf{W}_\alpha^S \mathbf{u}_j^Q)}, \tag{18}$$

$$\mathbf{c}_t = \sum_{i=1}^M \alpha_i^t \mathbf{u}_i^Q. \tag{19}$$

Here, t is a sentence index and i, j are question word indices. In the sentence modeling layer $\mathbf{v}_o^S$, $\mathbf{u}_t^S$ is used as input, and context vector $\mathbf{c}_t$ of the question-sentence matching layer is concatenated as $[\mathbf{u}_t^S; \mathbf{c}_t]$. The expression $[\mathbf{u}_t^S; \mathbf{c}_t]$ performs the element-wise product for the sigmoid-applied nonlinear gate layer $\mathbf{g}_t$. Finally, the sentence modeling $\mathbf{v}_o^S$ calculates the score along with question vector **q** in the output layer.

The sentence-passage matching layer ($S - P$ *match*) also uses the gated attention-based RNN to include the modeled sentence encoding hidden state $\mathbf{v}_i^S$ in the passage hidden state $\mathbf{u}_t^P$. Following this, one more abstraction is performed in modeling layer 1, and the equation is shown in the following. Now, the input of the passage modeling layer $\mathbf{v}_n^P$ is $\mathbf{g}_t \odot [\mathbf{u}_t^P; \mathbf{c}_t]$ with gated attention. In the following expression, t is a word index and i, j are sentence indices:

$$\mathbf{v}_t^P = BiSRU_{SP}\left(\mathbf{v}_{t-1}^P, \mathbf{g}_t \odot \left[\mathbf{u}_t^P; \mathbf{c}_t\right]\right), \tag{20}$$

$$\mathbf{g}_t = sigmoid\left(\mathbf{W}_g^P\left[\mathbf{u}_t^P; \mathbf{c}_t\right]\right), \tag{21}$$

$$\alpha_i^t = \frac{\exp(\mathbf{u}_t^P \mathbf{W}_\alpha^P \mathbf{v}_i^S)}{\sum_{j=1}^S \exp(\mathbf{u}_t^P \mathbf{W}_\alpha^P \mathbf{v}_j^S)}, \tag{22}$$

$$\mathbf{c}_t = \sum_{i=1}^S \alpha_i^t \mathbf{v}_i^S. \tag{23}$$

The self-matching layer applies self-attention to the passage encoding hidden state in modeling layer 2. The self-matching layer involves applying an attention mechanism to a given sequence

Table 1. MindsMRC Dataset Statistics

	Train Set	Dev Set	Test Set	Total
Number of Questions	141,424	8,054	8,053	157,531
Number of Passages	49,766	6,221	6,220	62,207

(i.e., itself) and using this as an input. The formulas for the modeling layer 2 ($\mathbf{h}_n^P$), where t, i, j are word indices, are expressed as follows:

$$\mathbf{h}_t^P = BiSRU_{SF}\left(\mathbf{h}_{t-1}^P, \mathbf{g}_t \odot \left[\mathbf{v}_t^P; \mathbf{c}_t\right]\right), \tag{24}$$

$$\mathbf{g}_t = sigmoid\left(\mathbf{W}_{g2}^P\left[\mathbf{v}_t^P; \mathbf{c}_t\right]\right), \tag{25}$$

$$\alpha_i^t = \frac{\exp(\mathbf{v}_t^P \mathbf{W}_h^P \mathbf{v}_i^P)}{\sum_{j=1}^N \exp(\mathbf{v}_t^P \mathbf{W}_h^P \mathbf{v}_j^P)}, \tag{26}$$

$$\mathbf{c}_t = \sum_{i=1}^N \alpha_i^t \mathbf{v}_i^P. \tag{27}$$

Based on the hierarchical pointer networks [14], the output layer points to the start (P_{start}) and end (P_{end}) positions of the answers corresponding to a question within a passage. The formulas for the out layer are expressed as follows:

$$P_{start}(i) \propto \exp\left(\mathbf{v}_{t,i}^S \mathbf{W}_s^S \mathbf{q}\right) * \exp\left(\mathbf{h}_i^P \mathbf{W}_s^P \mathbf{q}\right), \tag{28}$$

$$P_{end}(i) \propto \exp\left(\mathbf{v}_{t,i}^S \mathbf{W}_e^S \mathbf{q}\right) * \exp\left(\mathbf{h}_i^P \mathbf{W}_e^P \mathbf{q}\right), \tag{29}$$

where i is the word index and t is the sentence index of the sentence to which the i-th word belongs. Thus, S^3-NET calculates both the sentence and passage modeling information to find the position of the start and end of each answer span. First, to find the sentence closest to the question, we construct a sentence score vector by computing the sentence modeling hidden state $\mathbf{v}_t^S$ and the question vector $\mathbf{q}$ using bilinear sequence attention. Next, $\mathbf{h}_i^P$ modeled in modeling layer 2 and question vector $\mathbf{q}$ are calculated (also using bilinear sequence attention), and a passage score vector is created. The sentence and passage score vectors are multiplied by each other to find out the position (span) of the correct answer for each question.

4 EXPERIMENTS

4.1 Dataset

We used the SQuAD dataset [18] for English and SQuAD style MindsMRC dataset [15] for Korean. We performed cross-validation for our experiments. SQuAD is a reading comprehension dataset consisting of more than 23k passages and 100k questions manually annotated by crowdworkers on a set of Wikipedia articles. The passages come from approximately 500 Wikipedia articles. MindsMRC is a reading comprehension dataset for Korean, totally containing more than 62k passages and 157k questions posed by crowdworkers on sets of Wikipedia and Korean News articles. The MindsMRC statistics are listed in Table 1.

4.2 Implementation Details

In this work, we conducted the following experiments for MRC-QA using S^3-NET for SQuAD: The activation functions for the hidden and the attention layers were both tanh, and all of the RNN layers used BiSRU. The dropout (passage and question encoding and modeling layers) was fixed at 0.2, and the number of dimensions of the character embedding was set to 50. Furthermore, the

Table 2. Performance of S^3-NET Model with Other Models for SQuAD (dev, %)

Single Model	EM	F1
BiDAF [19]	67.7	77.3
FastQAExt [23]	70.3	78.5
DrQA [2]	69.5	78.8
Smarnet [3]	71.4	80.2
r-net [22]	72.3	80.6
S^3-NET (our)	71.5	80.7

word embedding dimension was set to 300 using GloVe word embedding [16], and the hidden layer dimension was set to 150. In addition, the character embedding used CNN, and the size of each filter was set to 30 by using a filter with sizes of 2, 3, 4, 5, and 6. The sentence representation also used CNN, the filter window size of 3, 4, 5, and 6 number of dimensions of each filter were set to 30. The encoder used a 5-stack hidden layer and a 2-stack modeling layer. The Adam algorithm [10] was used for optimization, the rate of which was set to 0.1. We did not use regularization. The size of the mini-batch was set to 32, and performance evaluation was performed with a development set for each epoch to obtain optimal models. EM and F1 were used as measures of performance [18]. EM measures the performance of equally matched words by comparing the correct answer with the system result, and F1 is the harmonic average of the precision and the recall of the system.

For the Korean MindsMRC dataset, hyper-parameters of S^3-NET were as follows: The activation functions, dropout, the setting of the character embedding (e.g., the number of dimensions of the character embedding, character embedding based on CNN, the size of each filter, filter with sizes), the setting of the sentence representation (e.g., based on CNN, filter window size, and the number of dimensions of each filter), the number of hidden layers of the encoder, the number of modeling layers, and Adam algorithm were the same as S^3-NET for SQuAD. The number of word embedding dimensions was 100, and the number of dimensions in the hidden layer was 128.

4.3 Experimental Results for the SQuAD Dataset (English)

We show the experimental results for the SQuAD dataset in Tables 2 through 5. Table 2 shows the performance comparison between our proposed S^3-NET and other systems for the SQuAD dataset. Experiments on the development set showed that our proposed model produced an 80.7% F1, which is superior to that of other existing models.

Table 3 shows the results of performing ablation experiments to measure the performance contribution of each feature for the SQuAD dataset. As shown in Table 3, F1 has the least influence, performing at 80.64% when the $f_{aligned}(q)$ feature was excluded. When the EM and token feature for passages were excluded, F1 showed a drop of 80.50%. When we excluded the $f_{aligned}(p)$ feature, the difference was −0.32% and the POS tag was −0.34% (similar to the results shown earlier). Furthermore, the difference was −0.49% when excluding character embedding. When NER was excluded, the performance decreased by −0.52%. When excluding the EM and token feature for the question, the performance deterioration was significantly larger than the ablation of the previous features (−0.85%), but all showed a performance drop of less than 1%. When excluding the hierarchical sentence module, a decrease of −1.90% was produced. Finally, when excluding all features such as the EM, token feature, aligned sentence embedding, POS tag, NER tag, and character embedding for both the question and passage, the difference was as large as 13.90%. We measured the t-test to ensure whether the incorporation of features and the hierarchical sentence module

Table 3. S^3-NET Ablation for SQuAD (dev, %)

Feature	**F1**	Δ
Our model	80.71	—
$-f_{aligned}(q)$	80.64	−0.07
$-(f_{exact\_match}(p) + f_{tf}(p))$	80.50	−0.21
$-f_{aligned}(p)$	80.39	−0.32
−POS tag	80.37	−0.34
−character CNN	80.22	−0.49
−NER tag	80.19	−0.52
$-(f_{exact\_match}(q) + f_{tf}(q))$	79.86	−0.85
−hierarchical sentence module	78.81	−1.90
−all features	66.81	−13.90

Table 4. Performance of the S^3-NET Model with Other Models for the SQuAD Test Set at Time of Writing (December 23, 2017) (%)

Single Model	**EM**	**F1**
LR Baseline [18]	40.40	51.00
BiDAF [19]	67.97	77.32
FastQAExt [23]	70.85	78.86
DrQA [2]	70.73	79.35
ReasoNet [20]	70.56	79.37
Mnemonic Reader [8]	71.00	80.15
Smarnet [3]	71.42	80.16
S^3-NET (our)	71.94	81.02
r-net [22]	81.39	88.17
BERT [6]	85.08	91.84
XLNet	89.90	95.08
Ensemble Model	**EM**	**F1**
BiDAF [19]	73.74	81.53
S^3-NET (our)	74.12	82.34
r-net [22]	84.00	90.15
BERT [6]	87.43	93.16
Human performance	82.30	91.22

brings in significant improvements. First, a result for S^3-NET and S^3-NET excluding all features is statistically significant with a p-value of 2.1235e-224 at a significance level of 0.05 (95% confidence level). Furthermore, the paired t-test result for the S^3-NET model and hierarchical sentence module removed is statistically significant with a p-value of 6.91e-11 at a significance level of 0.05. Thus, we can see that the proposed hierarchical encoding method is promising.

The official SQuAD leaderboard results are shown in Table 4 for the SQuAD dataset. The S^3-NET proposed in this article showed a somewhat lower performance compared to the recently updated models in the single model domain; however, our model outperformed the other models without the ELMo [17] and the BERT [6]. In the ensemble model, the performance was lower than that of the newly updated models, but the performance of F1 was approximately 0.81% higher than that

Table 5. Training Time of the S^3-NET Model with Other Models for SQuAD (Dev, Seconds Per Epoch)

Model	**Training Time**
BiDAF (our implementation)	1,020
DrQA [2]	191
r-net (our implementation)	1,071
S^3-NET (our)	420

Table 6. Performance of the S^3-NET Model with Other Models for the MindsMRC Dataset (%)

	Dev		**Test**	
Single Model	**EM**	**F1**	**EM**	**F1**
DrQA [2]	64.22	77.04	—	—
DrQA + BiSRU (our implementation)	64.99	77.98	—	—
BiDAF (our implementation)	66.00	78.89	—	—
BiDAF + Self-matching (our implementation)	66.31	78.38	—	—
DrQA + BiSRU + ($f_{exact\_match}(q) + f_{tf}(q)$) (our implementation)	67.38	79.90	—	—
S^3-NET (our)—all features	58.44	70.98	—	—
S^3-NET (our)—hierarchical sentence module	68.95	81.15	68.82	81.25
S^3-NET (our)	70.08	81.78	69.43	81.53
Ensemble Model	**EM**	**F1**	**EM**	**F1**
S^3-NET (our)	70.96	82.25	71.28	82.67

of the existing BiDAF. For the ensemble model, we simply choose the highest score of the answer among 10 single models that are exactly the same architecture except the random initial seed.

The training time comparison between our proposed S^3-NET and other systems is shown in Table 5 for the SQuAD dataset. All models were implemented in PyTorch. S^3-NET used SRU, and the others used GRU. All other hyper-parameters are the same. As a result, the training time of S^3-NET was 420 seconds, 2.19 times slower than DrQA, which is the basic model of ours but 1.92 and 2.01 times faster than BiDAF and r-net, respectively.

4.4 Experimental Results for the MindsMRC Dataset (Korean)

We show the results of the MindsMRC dataset in Korean for S^3-NET in Table 6. Table 6 shows the results of the comparison between S^3-NET and other models for MRC-QA (DrQA, DrQA + BiSRU, BiDAF, BiDAF + Self-matching). Here, DrQA + BiSRU, BiDAF, and BiDAF + Self-matching models are our implementations with PyTorch. The DrQA is the baseline of the experiment, RNN type is LSTM, and the number of stacks of the encoder layer and the modeling layers is 1 and 3, respectively. However, the other models apply SRU, and the number of the encoder and the modeling layers is 2 and 5, respectively. The size of the mini-batch was set to 32, and performance evaluation was performed with the development set for each epoch.

Experimental results show that the proposed S^3-NET in the developments set of Korean MindsMRC has the best performances of 70.08% for EM and 81.78% for F1 in the single model, and 70.96% for EM and 82.25% for F1 in the ensemble model. The ensemble model in the MindsMRC dataset also picked the highest score for the 10 single models with the same architecture except the initial random seed. In the case of the single model, DrQA + BiSRU with the EM feature of the question shows good performance with 67.38 for EM and 79.90% for F1 among the other compar-

Table 7. S^3-NET Ablation for the MindsMRC Dataset (Dev, %)

Feature	**F1**	Δ
Our model	81.78	—
$-f_{aligned}(q)$	80.72	−1.06
$-f_{aligned}(p)$	80.56	−1.22
$-(f_{exact\_match}(p) + f_{tf}(p))$	80.36	−1.42
$-(f_{exact\_match}(q) + f_{tf}(q))$	78.83	−2.95
−character CNN	78.71	−3.07
$-(f_{aligned} + f_{exact\_match} + f_{tf}$ + character CNN)	70.82	−10.96

ative models except for S^3-NET. S^3-NET shows better performances of +5.86% for EM and +4.74% for F1 than the baseline model DrQA. Our model, moreover, improves the performances by 5.09 to 2.7% for EM and 3.8 to 1.88% for F1 higher than the others such as DrQA + BiSRU, BiDAF, BiDAF + Self-matching, and DrQA + BiSRU + exact match feature for question. In S^3-NET, the performance difference is 10.80% when excluding all features such as the EM, token feature, aligned sentence embedding, POS tag, NER tag, and character embedding. The performance difference is 1.13% for EM and 0.63 for F1 when the hierarchical sentence module is excluded. The *t*-test result for S^3-NET and S^3-NET excluding all features is statistically significant with a *p*-value of 2.6119e-138 at a significance level of 0.05. The paired *t*-test result of S^3-NET and the model without a hierarchical sentence module is statistically significant with a *p*-value of 0.038 at a significance level of 0.05. It seems to be helpful to find the sentence containing the answer to the question using comprehending sentences of a single document composed of several sentences by the hierarchical sentence module. For the test set, the performance of the S^3-NET single model is 69.43% for EM and 81.53% for F1, and the ensemble model shows an improvement of +1.85% for EM and +1.14% for F1. Thus, we can see that the ensemble model performs better in the test set than the development set.

As listed in Table 7, all features contribute to the performance of our S^3-NET model for the MindsMRC dataset. Performance degradation of less than 2% is observed, with a difference of −1.06%, −1.22%, and −1.42%, when we remove the $f_{aligned}(q)$ feature, $f_{aligned}(p)$ feature, and the EM and token feature for passages, respectively. When we excluded the EM and token feature for the question, F1 showed a drop of 78.83% that is decreased by −2.95% from our proposed model, and thus this feature is more significant for question answering than previous features. The performance was 78.71% when excluding the character embedding, which shows a higher decrease than that for the previous features. This is why it was used to solve the unknown word problem when answering Korean questions. Finally, when we excluded the preceding features such as the EM, token feature, aligned sentence embedding, and character embedding for both the question and passage, the difference was as large as −10.96%.

4.5 Analysis

Our proposed S^3-NET encodes the input sequence using SRU, performs sentence matching, and calculates the output score based on the hierarchical structure. In this work, to verify the performance of S^3-NET for solving MRC-QA for the SQuAD and MindsMRC datasets, the following tasks are performed: (1) We measure the variation in performance for different RNN types, (2) we perform qualitative evaluation according to the presence or absence of the hierarchical module, and (3) we verify the performance of each model for various passage lengths.

4.5.1 Performance for RNN Types of S^3-NET on SQuAD and MindsMRC. We use SRU, which is faster than GRU or LSTM, because S^3-NET should encode the sentences included in the passage as

Table 8. Comparison of Performance of the S^3-NET Model with Other RNN Types on Dev Set

(a) Comparison of performance of the S^3-NET model with other RNN types for development of SQuAD (%, seconds per epoch)

RNN Type	Number of Layer Stacks			EM	F1	Time
	Question Encoder	Passage Encoder	Modeling Layer			
GRU	3	3	1	62.27	77.05	970
GRU	5	5	2	67.73	77.23	1,421
LSTM	3	3	1	66.50	76.12	1,090
LSTM	5	5	2	66.74	76.02	1,469
SRU	5	5	2	71.49	80.71	625

(b) Comparison of performance of the S^3-NET model with other RNN types for development of the MindsMRC dataset (%, seconds per epoch)

RNN Type	Number of Layer Stacks			EM	F1	Time
	Question Encoder	Passage Encoder	Modeling Layer			
GRU	3	3	1	65.61	77.93	2,058
GRU	5	5	2	64.91	77.11	2,667
LSTM	3	3	1	65.92	78.32	2,754
LSTM	5	5	2	65.66	77.92	3,628
SRU	5	5	2	70.08	81.78	1,630

well as the given questions and passage. Table 8 compares various RNN types for S^3-NET. The RNN types employed in the performance comparison are GRU and LSTM, and the hyper-parameters of S^3-NET used in the experiment are the same except for the number of layer stacks in Table 8. We set the number of layer stacks of the question and passage encoders to 3 and 5, and the number of stacks of the modeling layer to 1 and 2, for both GRU and LSTM. Table 7(a) and (b) are based on the SQuAD and MindsMRC datasets, respectively. Experimental results show the training time per epoch and the EM and F1 performances. Experimental results show that EM is 3.76% to 4.99% and F1 is 3.48% to 4.69% better than the corresponding values for GRU and LSTM in the case of (a) using SRU with EM of 71.49% and F1 of 80.71% for the development set for SQuAD. The training time per epoch is 625 seconds, which is 1.55 to 2.35 times faster than that of other RNN types. As shown in Table 8(b), the best performances are obtained when SRU is used, with a difference of 4.42% to 5.17% in EM and 3.46% to 4.67% in F1 compared to the corresponding values for GRU and LSTM. In this case, the training time per epoch is 1,630 seconds, which is 1.26 to 2.23 times faster than that of other RNN types. Accordingly, we confirmed that SRU is the most suitable architecture for the S^3-NET model proposed in this article. This is because it removes the recurrence of gates to create a structure that can be processed in parallel; furthermore, it processes history information by using the recurrent cell and preserves low-level information by using the highway network [11]. Thus, it shows good performance.

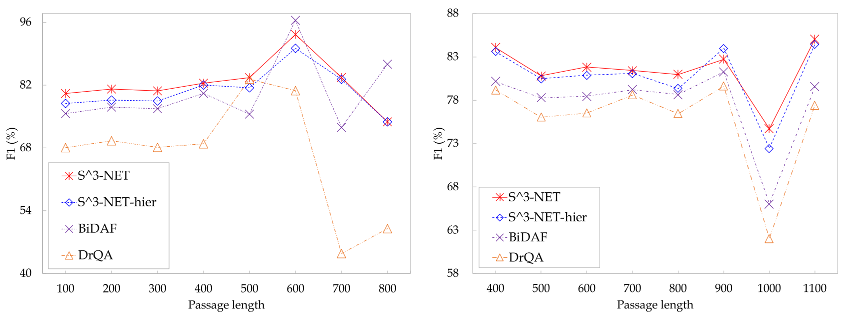
4.5.2 *Result Examples for Presence or Absence of the Hierarchical Module.* Table 9 shows the effect of hierarchical model extension when solving various machine reading tasks. The sentences in the table are excerpts containing a correct answer and an incorrect answer in a passage. The correct answer is indicated in bold, and the incorrect answer is underlined. The models used for the result comparison are S^3-NET and S^3-NET-hier, which does not include the hierarchical module. Table 9 shows the effect of inclusion of the hierarchical module. The total number of sentences in the first example of SQuAD is four, and the second sentence has the correct answer. In the case of S^3-NET-hier, the error result "National Football League" included

Table 9. Example of Solved Result for SQuAD and MindsMRC

Example **SQuAD**	**S^3-NET-hier**	**S^3-NET**	**Correct Answer**
Q: Which NFL team won Super Bowl 50? Sen.: Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion **Denver Broncos** defeated the National Football Conference (NFC) champion Carolina Panthers 24 to earn their third Super Bowl title.	National Football League	Denver Broncos	Denver Broncos
Q: What is the 16th century known as the start of? Sen.: In the 16th century, the beginning of **the historical era**, the region was inhabited by the Mocama, a coastal subgroup of the Timucua people. At the time of contact with Europeans, all Mocama villages in present-day Jacksonville were part of the powerful chiefdom known as the Saturiwa, centered around the mouth of the St. Johns River.	Saturiwa	the historical era	the historical era
MindsMRC			
Q: 프레디 프리먼이 어디 팀이야? Sen.: 강정호(29 · 피츠버그 파이리츠)의 뜨거운 방망이가 9월 말에도 식을 줄 모른다. 중요한 건 경쟁자의 성적인데, 9월 타율 0.405를 기록 중인 프레디 프리먼(**애틀랜타 브레이브스**)과 홈런 7개를 때린 맷 켐프(애틀랜타 브레이브스) 등이 후보로 꼽힌다.	피츠버그 파이리츠	애틀랜타 브레이브스	애틀랜타 브레이브스
Q: 엘튼 존이 친구들과 만든 팀 이름이 뭐야? Sen.: 엘튼 존은 양아버지의 이름 'Fred'의 철자를 거꾸로 바꾸어서 그를 '더프(Derf)'라는 애칭으로 불렀다. 1961년, 엘튼 존은 자신의 학교 친구들과 함께 '**콜벳스**'를 결성했다. 콜벳스는 피너의 교회나 각종 클럽에서 공연을 열어 주로 로큰롤과 블루스를 연주했지만 몇차례의 공연을 끝으로 1962년 봄에 해산했다.	더프	콜벳스	콜벳스

in the first sentence was provided as output, whereas S^3-NET, which includes the hierarchical structure, provided the correct answer contained in the second sentence. The second example of SQuAD has five sentences, and the correct answer is included in the third sentence. S^3-NET-hier again provided an incorrect answer as output, whereas S^3-NET provided the correct answer. In the Korean MindsMRC dataset as well, the effect of the hierarchical structure can be observed. The first example of MindsMRC has 24 sentences, and the sentence with the correct answer is the 23rd in the passage. From the experiment, when the hierarchical structure is not used, 피츠버그 파이리츠, which is included in the first sentence in the passage, is provided as output, whereas when the hierarchical structure is used, the correct answer is provided as output. The second example of MindsMRC has five sentences, and the correct answer is in the fourth sentence. The S^3-NET-hier model again provides the incorrect answer contained in the first sentence as output, whereas S^3-NET provides the correct answer in the fourth sentence as output. Thus, the method of using sentence information together facilitates machine reading comprehension.

4.5.3 Performance over Various Passage Length. We compare the performances of the S^3-NET model and other models according to various passage lengths as shown in Figure 3. The models used for performance comparison are DrQA, BiDAF, S^3-NET-hier (which does not have the hierarchical structure), and S^3-NET. Figure 3(a) and (b) show the results for the SQuAD and MindsMRC datasets, respectively. In Figure 3(a), the performance of BiDAF is the best in the passage length ranges of 500 to 600 and 700 to 800, but over the remaining ranges, S^3-NET shows good overall performance. In Figure 3(b), the S^3-NET-hier model shows high performance in the passage length range of 800 to 900, but S^3-NET shows the best performance over the remaining ranges, including the longest passage range of 1,000 to 1,100.



(a) F1 over various passage lengths on SQuAD

(b) F1 over various passage lengths on MindsMRC

Fig. 3. Score over various passage lengths on dev set.

5 CONCLUSION

In this article, we proposed S^3-NET, a model that has a robust hierarchical structure for long sentences. Experimental results on the SQuAD dataset show that the proposed method produced 71.49% for EM and 80.71% for F1 when used as a single model. In addition, S^3-NET produced 73.6% for EM and 81.9% for F1 when used as an ensemble model in a development set. Moreover, the performances of EM 71.91% and F1 81.02% were produced when S^3-NET was used as a single model in a test set. Finally, the performances of EM 74.12% and F1 82.34% were produced when S^3-NET was used as an ensemble model in a test set. For the Korean MindsMRC dataset, experimental results show that our model achieved 69.43% for EM and 81.53% for F1 of the single model and 71.28% for EM and 82.67% for F1 of the ensemble model.

In addition, we used CNN to perform sentence encoding. Unlike RNN, local dependency can be performed with CNN that is encoded by various feature maps. Thus, high-coherency coding can be performed for each sentence. In addition, because long-term sequence encoding is performed using RNN in the passage encoder, the difference was observed.

In future research, we will perform ELMo-based ensemble modeling [17].

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473.

[2] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. arXiv:1704.00051.

[3] Zheqian Chen, Rongqin Yang, Bin Cao, Zhou Zhao, Deng Cai, and Xiaofei He. 2017. Smarnet: Teaching machines to read and comprehend like human. arXiv:1710.02772.

[4] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv:1406.1078.

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805.

[7] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The Goldilocks principle: Reading children's books with explicit memory representations. arXiv:1511.02301.

[8] Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2017. Reinforced mnemonic reader for machine reading comprehension. arXiv:1705.02798.

[9] Yoon Kim. 2014. Convolutional neural networks for sentence classification. arXiv:1408.5882.

[10] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv:1412.6980.

[11] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. arxiv:cs.LG/1505.00387.

[12] Tao Lei and Yu Zhang. 2017. Training RNNs as fast as CNNs. arXiv:1709.02755.

[13] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. arXiv:1611.09268.

[14] Cheoneum Park and Changki Lee. 2017. Coreference resolution using hierarchical pointer networks. *KIISE Transaction on Computing Practices* 23, 9 (2017), 542–549.

[15] Cheoneum Park, Changki Lee, Lynn Hong, Yigyu Hwang, Taejoon Yoo, Jaeyong Jang, Yunki Hong, Kyung-Hoon Bae, and Hyun-Ki Kim. 2019. S2-Net: Machine reading comprehension with SRU-based self-matching network. *ETRI Journal* 41, 3 (2019), 371–382.

[16] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. 1532–1543.

[17] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. arXiv:1802.05365.

[18] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. arXiv:1606.05250.

[19] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. arXiv:1611.01603.

[20] Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2017. ReasoNet: Learning to stop reading in machine comprehension. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, 1047–1055.

[21] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*. 2692–2700.

[22] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 189–198.

[23] Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017. Making neural QA as simple as possible but not simpler. arXiv:1703.04816.

Received August 2018; revised July 2019; accepted September 2019